

Formulario pruebas unitarias proyecto ADIIC

Formato Optimizado de Prueba Unitaria – PU01

| | |
|--------------------------------|--|
| ID Prueba | PU01 |
| Épica | EPICA 1 Inicio de sesión y autenticación |
| Función probada | autenticarUsuario en authController.js |
| Descripción | Autenticación de usuario con credenciales válidas. |
| Entradas de prueba | {email: 'usuario@example.com', password: 'secret123'} |
| Criterios de aceptación | Debe devolver HTTP 200 OK, token JWT válido y los datos del usuario autenticado. |
| Resultado esperado | El servidor debe responder con HTTP 200 y devolver un token JWT y los datos básicos del usuario. |
| Resultado obtenido | El servidor respondió con HTTP 200 y devolvió un token JWT válido más los datos del usuario (id, nombre, email). |
| Estado de la prueba | ✓ Exitosa |
| Observaciones | La lógica de autenticación y generación de token funciona correctamente en el controlador y en las rutas protegidas. |

| | |
|------------------------|---|
| ID Prueba | PU01 |
| Épica | EPICA 1 – Inicio de sesión y autenticación |
| Función probada | autenticarUsuario en authController.js |
| Descripción | Autenticación de usuario con contraseña incorrecta. |

| | |
|--------------------------------|---|
| Entradas de prueba | {email: 'usuario@example.com', password: 'clave_incorrecta' } |
| Criterios de aceptación | Debe devolver HTTP 401 Unauthorized y un mensaje de error indicando que la contraseña es incorrecta. |
| Resultado esperado | El servidor debe responder con HTTP 401 y mensaje de error de credenciales inválidas. |
| Resultado obtenido | El servidor respondió con HTTP 401 y mostró el mensaje 'La contraseña es incorrecta'. |
| Estado de la prueba | ✗ Fallida |
| Observaciones | El sistema detecta correctamente contraseñas inválidas y no permite el acceso, cumpliendo con las políticas de seguridad. |

Formato Optimizado de Prueba Unitaria – PU02


| | |
|--------------------------------|--|
| ID Prueba | PU02 |
| Épica | EPICA 2 Gestión de usuarios |
| Función probada | crearUsuario en usuariosController.js |
| Descripción | Registro exitoso de usuario con datos válidos. |
| Entradas de prueba | {nombre: 'Juan Pérez', email: 'juan@example.com', rol: 'regular', password: 'clave123', confirmar: 'clave123' } |
| Criterios de aceptación | El usuario debe ser registrado correctamente y debe retornarse HTTP 201 Created con los datos básicos del usuario. |

| | |
|--------------------------------|---|
| Resultado esperado | El servidor debe responder con HTTP 201 y devolver el objeto del usuario creado. |
| Resultado obtenido | El servidor respondió con HTTP 201 y devolvió los datos del nuevo usuario sin incluir el password. |
| Estado de la prueba | ✓ Exitosa |
| Observaciones | El registro de usuario se realiza correctamente con validación de campos, encriptación de contraseña y respuesta clara del backend. |
| ID Prueba | PU02 |
| Épica | EPICA 2 Gestión de usuarios |
| Función probada | crearUsuario en usuariosController.js |
| Descripción | Intento de registrar un usuario con un email ya existente. |
| Entradas de prueba | {nombre: 'Juan Pérez', email: 'juan@example.com', rol: 'regular', password: 'clave123', confirmar: 'clave123' } |
| Criterios de aceptación | Debe rechazarse el registro y devolverse HTTP 400 con un mensaje de que el usuario ya existe. |
| Resultado esperado | El servidor debe responder con HTTP 400 y el mensaje 'El usuario ya existe'. |
| Resultado obtenido | El servidor respondió con HTTP 400 y mostró el mensaje 'El usuario ya existe'. |
| Estado de la prueba | ✗ Fallida |

| | |
|----------------------|---|
| Observaciones | La validación de duplicidad funciona correctamente al detectar que el email ya está registrado en la base de datos. |
|----------------------|---|

Formato Optimizado de Prueba Unitaria – PU03

| | |
|--------------------------------|--|
| ID Prueba | PU03 |
| Épica | EPICA 3 – Gestión de categorías |
| Función probada | crearCategoria en categoriasController.js |
| Descripción | Creación exitosa de una categoría con nombre e imagen válidos. |
| Entradas de prueba | {nombre: 'Tecnología', imagen: 'https://miweb.com/img/tecnologia.png'} |
| Criterios de aceptación | La categoría debe ser creada correctamente, retornando HTTP 201 Created y el objeto de la categoría. |
| Resultado esperado | El servidor debe responder con HTTP 201 y retornar el objeto de la categoría creada. |
| Resultado obtenido | El servidor respondió con HTTP 201 y devolvió correctamente el objeto con nombre e imagen. |
| Estado de la prueba | ✓ Exitosa |
| Observaciones | La creación de la categoría se realizó exitosamente validando los campos requeridos y almacenando correctamente en la base de datos. |

| | |
|--------------------------------|--|
| ID Prueba | PU03 |
| Épica | EPICA 3 – Gestión de categorías |
| Función probada | crearCategoria en categoriasController.js |
| Descripción | Intento de crear una categoría dejando el campo 'nombre' vacío. |
| Entradas de prueba | {nombre: "", imagen: 'https://miweb.com/img/tecnologia.png' } |
| Criterios de aceptación | No debe crearse la categoría si algún campo está vacío. Debe retornarse HTTP 400 con un mensaje indicando que todos los campos son obligatorios. |
| Resultado esperado | El servidor debe responder con HTTP 400 y mostrar el mensaje 'Todos los campos son obligatorios'. |
| Resultado obtenido | El servidor respondió con HTTP 400 y mostró el mensaje 'Todos los campos son obligatorios'. |
| Estado de la prueba |  Fallida |
| Observaciones | El backend valida correctamente los campos requeridos y rechaza el registro con campos vacíos, manteniendo la integridad de los datos. |

Formato Optimizado de Prueba Unitaria – PU04

| | |
|------------------|--------------------------------|
| ID Prueba | PU04 |
| Épica | EPICA 4 – Gestión de productos |

| | |
|--------------------------------|--|
| Función probada | crearProducto en productosController.js |
| Descripción | Registro de producto con datos válidos. |
| Entradas de prueba | {referencia: '445566', nombre: 'Teclado', stock: 10, precio: 50000} |
| Criterios de aceptación | El producto se crea exitosamente con datos válidos. Retorna HTTP 201 Created y el objeto del producto. |
| Resultado esperado | El servidor debe responder con HTTP 201 y devolver el objeto completo del producto registrado. |
| Resultado obtenido | El servidor respondió correctamente con HTTP 201 y devolvió un objeto de producto con ID, nombre, stock y precio. |
| Estado de la prueba | ✓ Exitosa |
| Observaciones | La validación de stock y demás campos se ejecuta correctamente tanto en el modelo Mongoose como en el controlador. |

| | |
|---------------------------|--|
| ID Prueba | PU04 |
| Épica | EPICA 4 – Gestión de productos |
| Función probada | crearProducto en productosController.js |
| Descripción | Registro de producto con stock negativo. |
| Entradas de prueba | {referencia: '445566', nombre: 'Teclado', stock: -1, precio: 50000} |

| | |
|--------------------------------|---|
| Criterios de aceptación | El stock no puede ser negativo. Debe mostrarse un mensaje de error y retornar HTTP 400 Bad Request. |
| Resultado esperado | HTTP 400 y mensaje de validación. |
| Resultado obtenido | HTTP 400 – El producto no fue registrado, ya que cuenta con stock negativo |
| Estado de la prueba | ✗ Fallida |
| Observaciones | Falta validación lógica en el backend o en el modelo. Se recomienda agregar verificación de stock ≥ 0 en el esquema Mongoose o middleware. |

Formato Optimizado de Prueba Unitaria – PU05

| | |
|--------------------------------|--|
| ID Prueba | PU05 |
| Épica | EPICA 5 – Cabeceras para facturas |
| Función probada | crearCabecera en cabeceraController.js |
| Descripción | Creación exitosa de una cabecera con todos los campos válidos. |
| Entradas de prueba | {local: 'Sucursal Norte', nit: '123456789', direccion: 'Calle 10 #5-23', telefono: '3001234567', email: 'norte@local.com'} |
| Criterios de aceptación | Debe crearse la cabecera correctamente si todos los campos están completos y válidos. Se debe retornar HTTP 201 con el objeto de la cabecera creada. |

| | |
|--------------------------------|--|
| Resultado esperado | El servidor debe responder con HTTP 201 y el objeto completo de la cabecera registrada. |
| Resultado obtenido | El servidor respondió con HTTP 201 y devolvió correctamente el objeto de la cabecera creada. |
| Estado de la prueba | ✓ Exitosa |
| Observaciones | La validación y almacenamiento de los campos funciona correctamente, generando una respuesta clara y precisa del backend. |
| | |
| ID Prueba | PU05 |
| Épica | EPICA 5 – Cabeceras para facturas |
| Función probada | crearCabecera en cabeceraController.js |
| Descripción | Intento de crear una cabecera dejando vacío el campo 'email'. |
| Entradas de prueba | {local: 'Sucursal Norte', nit: '123456789', direccion: 'Calle 10 #5-23', telefono: '3001234567', email: ''} |
| Criterios de aceptación | Debe rechazarse el registro si algún campo está vacío, retornando HTTP 400 con el mensaje 'Todos los campos son obligatorios'. |
| Resultado esperado | El servidor debe responder con HTTP 400 y mensaje 'Todos los campos son obligatorios'. |
| Resultado obtenido | El servidor respondió con HTTP 400 y mostró el mensaje 'Todos los campos son obligatorios'. |

| | |
|----------------------------|---|
| Estado de la prueba | ✗ Fallida |
| Observaciones | El sistema valida correctamente que ningún campo obligatorio quede vacío, y muestra un mensaje de error informativo al usuario. |

Formato Optimizado de Prueba Unitaria – PU06

| | |
|--------------------------------|---|
| ID Prueba | PU06 |
| Épica | EPICA 6 – Gestión de facturas |
| Función probada | ingresarFactura en facturaController.js |
| Descripción | Creación exitosa de una factura completa con cabecera, datos de cliente y detalle de productos. |
| Entradas de prueba | {cabecera: 'cab001', cliente: {nombre: 'Carlos Gómez', nit: '900123456', direccion: 'Av. Siempre Viva 742', telefono: '3101234567', ciudad: 'Bogotá' }, productos: [{referencia: '112233', nombre: 'Mouse', cantidad: 2, precio: 40000}]} } |
| Criterios de aceptación | La factura se registra correctamente retornando HTTP 201 Created y los datos completos: cabecera, cliente, detalle, subtotal, IVA y total. |
| Resultado esperado | El servidor responde con HTTP 201 y devuelve el objeto de la factura creada con importes calculados. |


| | |
|--------------------------------|---|
| Resultado obtenido | El servidor respondió con HTTP 201 e incluyó el objeto de la factura con cabecera, cliente, productos y campos monetarios en formato decimal. |
| Estado de la prueba | ✓ Exitosa |
| Observaciones | Los campos monetarios devueltos como <code>{'\$numberDecimal': 'valor'}</code> se formatean correctamente en el frontend; cálculos y almacenamiento funcionan sin errores. |
| ID Prueba | PU06 |
| Épica | EPICA 6– Gestión de facturas |
| Función probada | ingresarFactura en facturaController.js |
| Descripción | Intento de crear una factura dejando vacío el campo 'nombre' del cliente. |
| Entradas de prueba | <code>{cabecera: 'cab001', cliente: {nombre: "", nit: '900123456', direccion: 'Av. Siempre Viva 742', telefono: '3101234567', ciudad: 'Bogotá'}, productos: [{referencia: '112233', nombre: 'Mouse', cantidad: 2, precio: 40000}]} }</code> |
| Criterios de aceptación | Debe rechazarse la creación de la factura si algún campo obligatorio está vacío. Debe retornar HTTP 400 con el mensaje 'Todos los campos son obligatorios'. |

| | |
|----------------------------|---|
| Resultado esperado | El servidor debe responder con HTTP 400 y el mensaje 'Todos los campos son obligatorios'. |
| Resultado obtenido | El servidor respondió con HTTP 400 y mostró el mensaje 'Todos los campos son obligatorios'. |
| Estado de la prueba | ✖ Fallida |
| Observaciones | El backend valida correctamente los campos obligatorios y evita registrar facturas incompletas, manteniendo la integridad de los datos. |

Formato Optimizado de Prueba Unitaria – PU07

| | |
|--------------------------------|--|
| ID Prueba | PU07 |
| Épica | EPICA 7 – Gestión de clientes |
| Función probada | ingresarCliente en clienteController.js |
| Descripción | Registro exitoso de un cliente con todos los campos obligatorios correctamente diligenciados. |
| Entradas de prueba | {nombre: 'Richard Pardo Cardona', nit: '9737854', direccion: 'Calle 14 N. 16-20', ciudad: 'Armenia Quindio', telefono: '3105357317'} |
| Criterios de aceptación | El cliente debe registrarse exitosamente. El backend debe retornar HTTP 201 y el objeto del cliente creado. |
| Resultado esperado | El servidor responde con HTTP 201 y retorna el objeto con los datos del cliente registrado. |

| | |
|--------------------------------|--|
| Resultado obtenido | El servidor respondió con HTTP 201 y devolvió el objeto del cliente incluyendo su ID, nombre, NIT, dirección, ciudad y teléfono. |
| Estado de la prueba | ✓ Exitosa |
| Observaciones | El sistema valida correctamente los campos obligatorios y almacena los datos del cliente de manera adecuada. |
| ID Prueba | PU07 |
| Épica | EPICA 7 – Gestión de clientes |
| Función probada | ingresarCliente en clienteController.js |
| Descripción | Intento de registrar un cliente dejando vacío el campo 'nombre'. |
| Entradas de prueba | {nombre: "", nit: '9737854', direccion: 'Calle 14 N. 16-20', ciudad: 'Armenia Quindio', telefono: '3105357317'} |
| Criterios de aceptación | No debe permitirse registrar clientes si hay campos obligatorios vacíos. Debe retornarse HTTP 400 y mensaje 'Todos los campos son obligatorios'. |
| Resultado esperado | El servidor debe responder con HTTP 400 y el mensaje 'Todos los campos son obligatorios'. |
| Resultado obtenido | El servidor respondió con HTTP 400 y mostró el mensaje correspondiente a campos obligatorios faltantes. |

| | |
|----------------------------|---|
| Estado de la prueba |  Fallida |
| Observaciones | El backend valida correctamente la presencia de todos los campos requeridos y rechaza entradas incompletas. |