🔄 DIAGRAMA DE COLABORACIÓN: CREAR CATEGORÍA

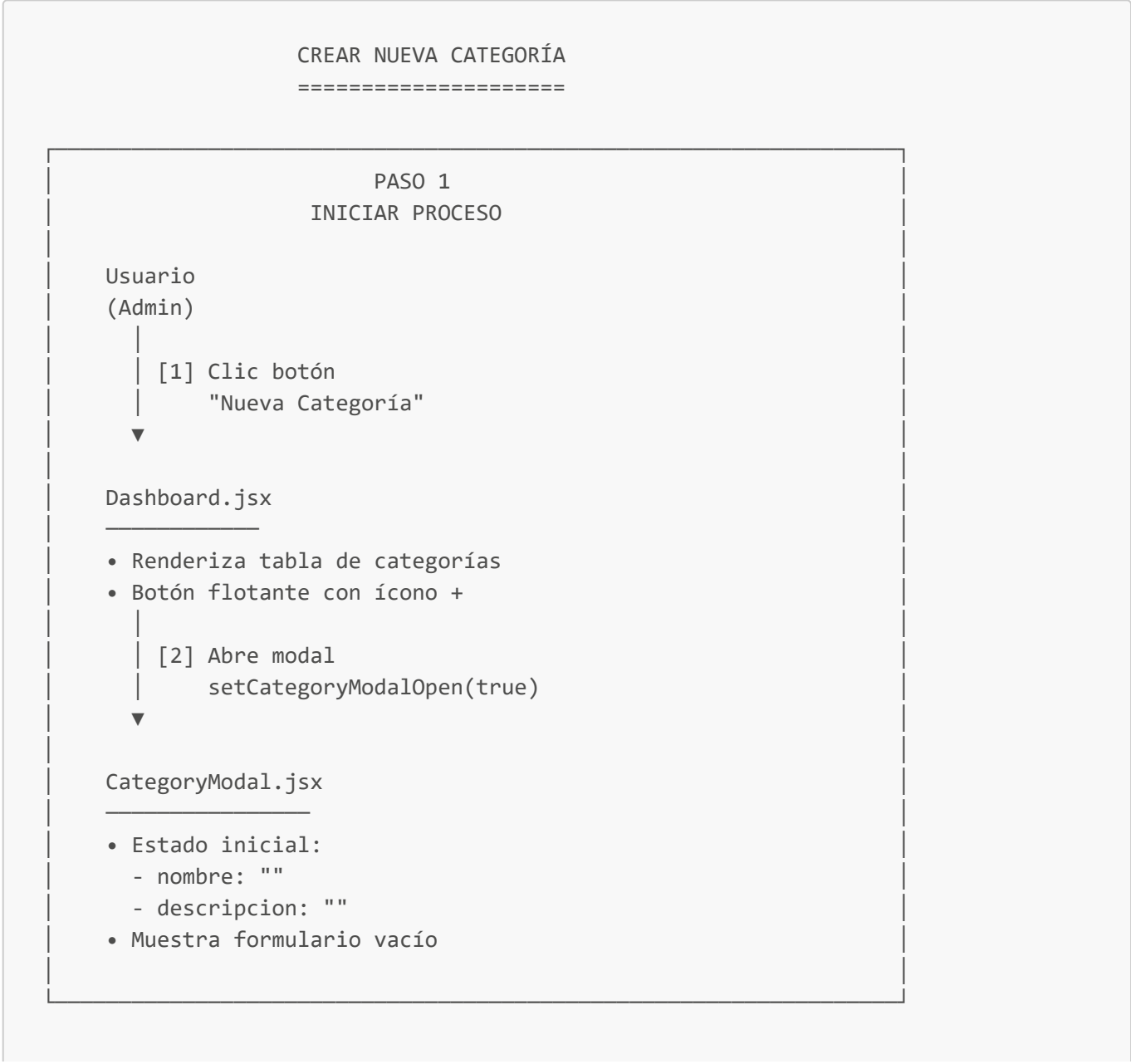
Sistema de Inventario - Flujo Detallado

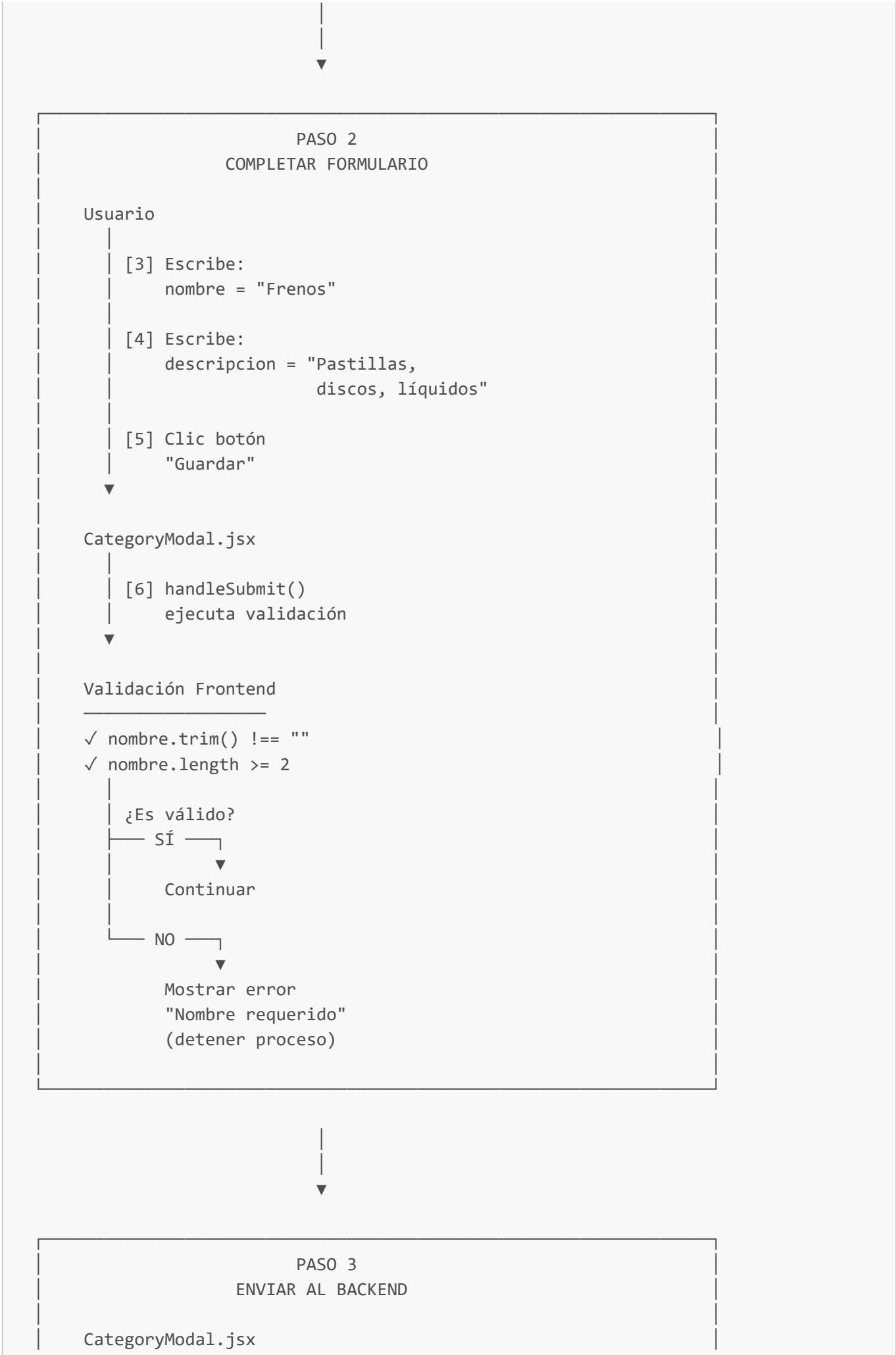
📋 DESCRIPCIÓN GENERAL

Este diagrama muestra el flujo completo cuando un usuario crea una nueva categoría en el sistema.

Duración estimada: 2-3 segundos
Actores: Administrador o Vendedor
Resultado: Nueva categoría registrada en la base de datos

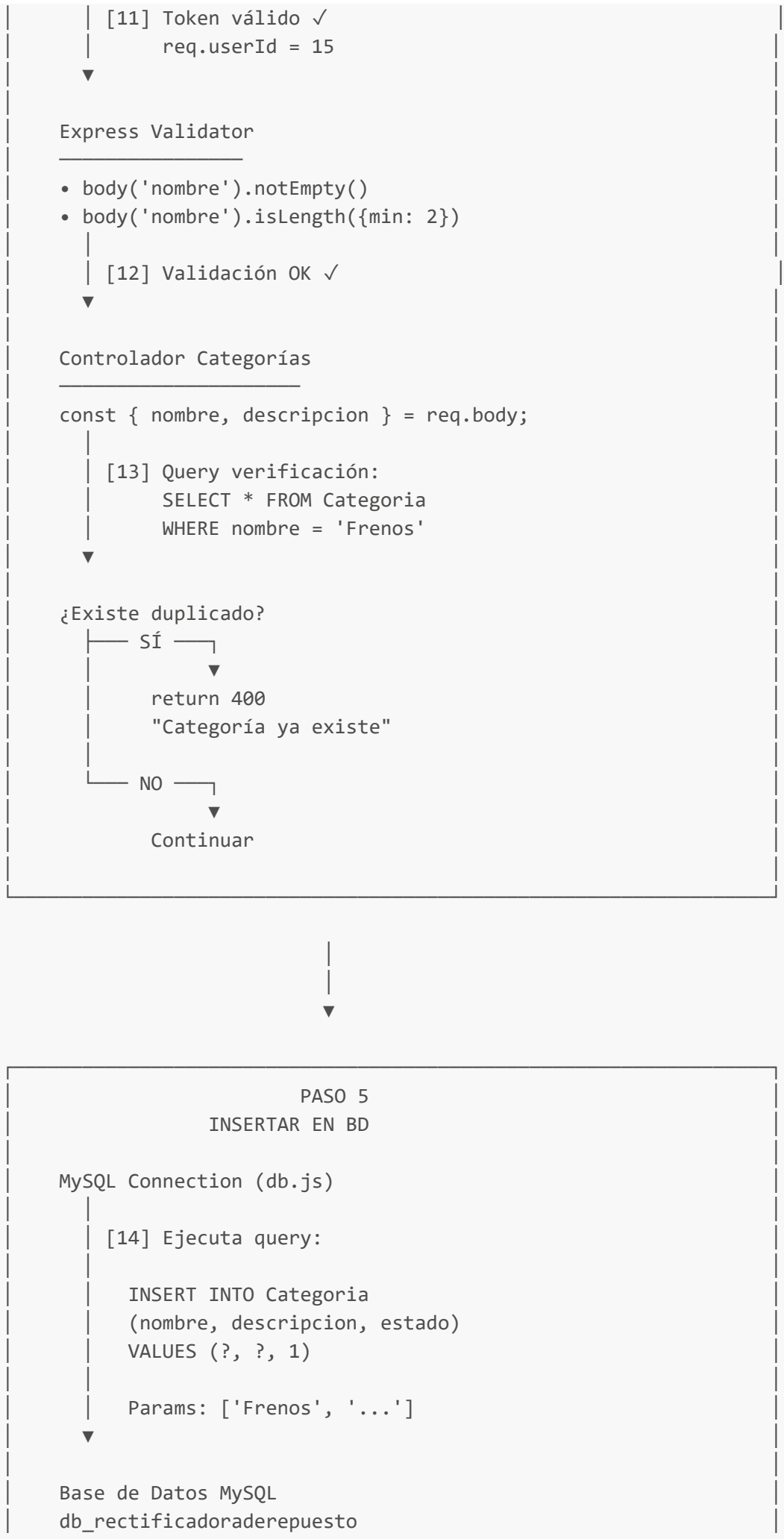
🎯 DIAGRAMA MEJORADO (Sin Cruces)




PASO 2
COMPLETAR FORMULARIO
Usuario
[3] Escribe:
nombre = "Frenos"
[4] Escribe:
descripcion = "Pastillas,
discos, líquidos"
[5] Clic botón
"Guardar"
CategoryModal.jsx
[6] handleSubmit()
ejecuta validación
Validación Frontend
✓ nombre.trim() !== ""
✓ nombre.length >= 2
¿Es válido?
SÍ
Continuar
NO
Mostrar error
"Nombre requerido"
(detener proceso)
PASO 3
ENVIAR AL BACKEND
CategoryModal.jsx


[7] Construye objeto:
{
nombre: "Frenos",
descripcion: "..."
}
[8] api.post()
URL: /api/categorias
api.js (Axios)
• Interceptor agrega:
Authorization: Bearer <JWT_TOKEN>
[9] HTTP POST
http://localhost:5000/api/categorias
Headers: {
Authorization: "Bearer eyJ..."
Content-Type: "application/json"
}
Body: {
"nombre": "Frenos",
"descripcion": "..."
}
RED (INTERNET/LOCALHOST)


PASO 4
PROCESAMIENTO BACKEND
Express Server (index.js)
[10] Recibe POST
/api/categorias
Middleware: verifyToken
• Extrae token del header
• jwt.verify(token, SECRET)
• Decodifica userId


[11] Token válido ✓
req.userId = 15
Express Validator
• body('nombre').notEmpty()
• body('nombre').isLength({min: 2})
[12] Validación OK ✓
Controlador Categorías
const { nombre, descripcion } = req.body;
[13] Query verificación:
SELECT * FROM Categoria
WHERE nombre = 'Frenos'
¿Existe duplicado?
SÍ
return 400
"Categoría ya existe"
NO
Continuar
PASO 5
INSERTAR EN BD
MySQL Connection (db.js)
[14] Ejecuta query:
INSERT INTO Categoria
(nombre, descripcion, estado)
VALUES (?, ?, 1)
Params: ['Frenos', '...']
Base de Datos MySQL
db_rectificadoraderepuesto

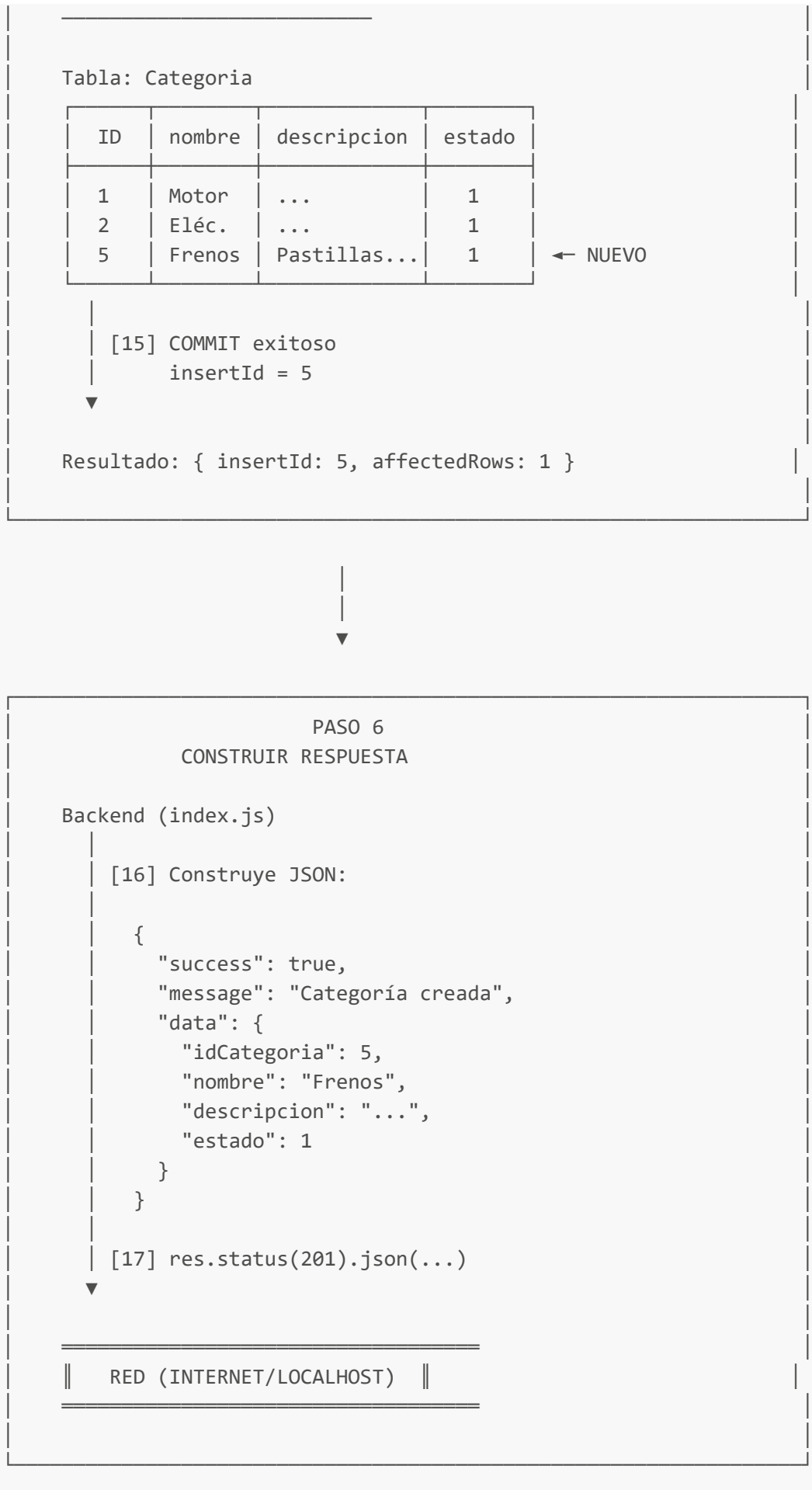
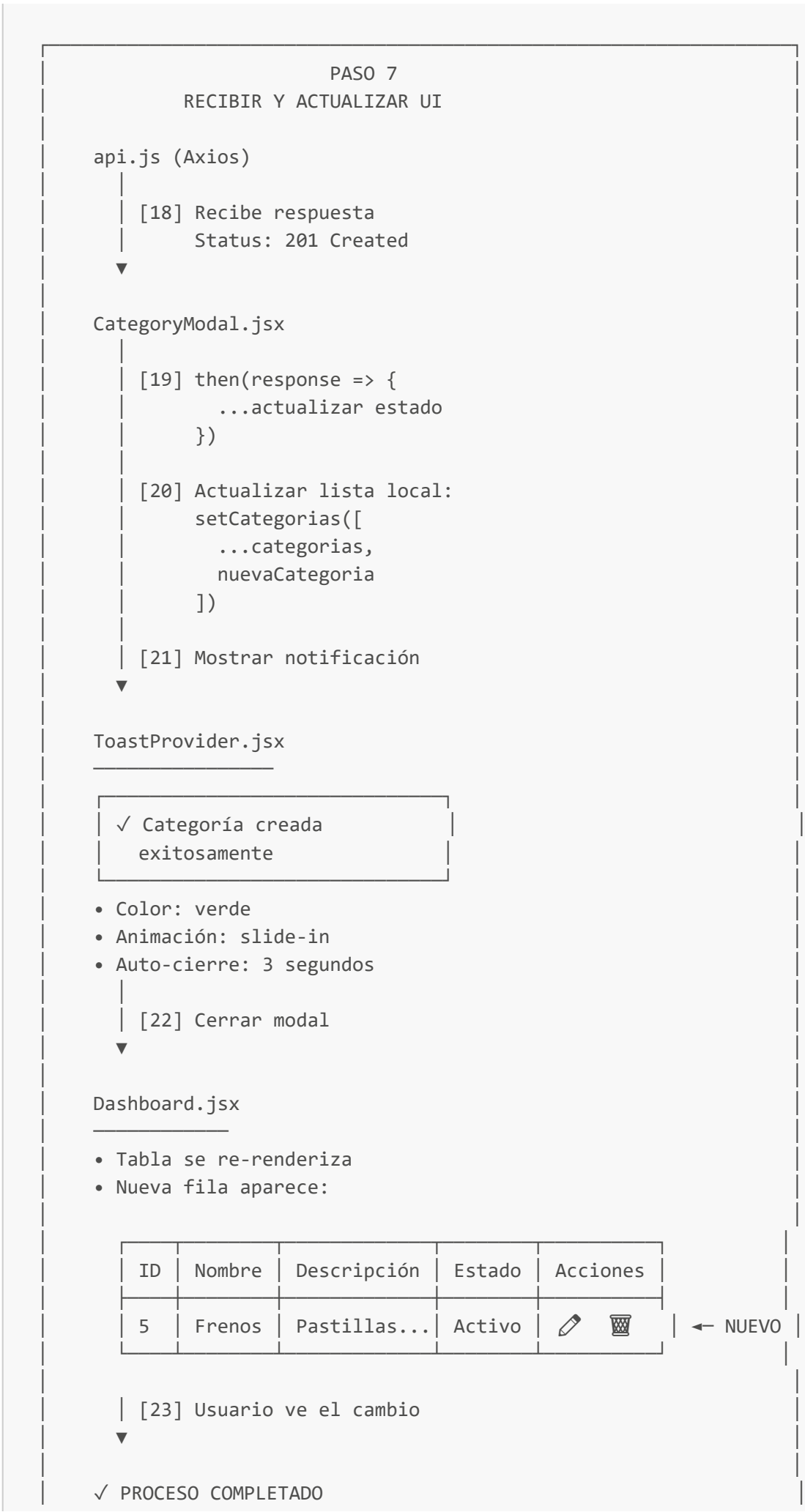


Tabla: Categoria
ID | nombre | descripcion | estado
1 | Motor | ... | 1
2 | Eléc. | ... | 1
5 | Frenos | Pastillas... | 1 ← NUEVO
[15] COMMIT exitoso
insertId = 5
Resultado: { insertId: 5, affectedRows: 1 }
PASO 6
CONSTRUIR RESPUESTA
Backend (index.js)
[16] Construye JSON:
{
"success": true,
"message": "Categoría creada",
"data": {
"idCategoria": 5,
"nombre": "Frenos",
"descripcion": "...",
"estado": 1
}
}
[17] res.status(201).json(...)
RED (INTERNET/LOCALHOST)


PASO 7
RECIBIR Y ACTUALIZAR UI
api.js (Axios)
[18] Recibe respuesta
Status: 201 Created
CategoryModal.jsx
[19] then(response => {
...actualizar estado
})
[20] Actualizar lista local:
setCategorias([
...categorias,
nuevaCategoria
])
[21] Mostrar notificación
ToastProvider.jsx
✓ Categoría creada exitosamente
• Color: verde
• Animación: slide-in
• Auto-cierre: 3 segundos
[22] Cerrar modal
Dashboard.jsx
• Tabla se re-renderiza
• Nueva fila aparece:
ID
Nombre
Descripción
Estado
Acciones
5
Frenos
Pastillas...
Activo
← NUEVO
[23] Usuario ve el cambio
✓ PROCESO COMPLETADO

```
│                                                                    │
└────────────────────────────────────────────────────────────────────┘
```

RESUMEN EJECUTIVO

Paso	Componente	Acción Principal	Tiempo
1	Dashboard → CategoryModal	Abrir formulario	0.1s
2	CategoryModal	Usuario completa datos	Variable
3	api.js	Enviar POST con JWT	0.1s
4	Backend (index.js)	Validar token y datos	0.2s
5	MySQL	Insertar registro	0.3s
6	Backend	Construir respuesta JSON	0.1s
7	CategoryModal → Dashboard	Actualizar UI	0.2s

Tiempo total: ~1-2 segundos (sin contar entrada de usuario)

VALIDACIONES EN CADA PASO

Frontend (CategoryModal.jsx):

```
// Validación antes de enviar
if (!nombre.trim()) {
  setError('El nombre es requerido');
  return;
}
if (nombre.length < 2) {
  setError('Mínimo 2 caracteres');
  return;
}
```

Backend (index.js):

```
// Express Validator
body('nombre')
  .notEmpty().withMessage('Nombre requerido')
  .isLength({ min: 2 }).withMessage('Mínimo 2 caracteres')
  .trim()

// Verificación de duplicados
const [existing] = await db.query(
  'SELECT idCategoria FROM Categoria WHERE nombre = ?',
```

```
    [nombre]
  );
  if (existing.length > 0) {
    return res.status(400).json({
      success: false,
      message: 'La categoría ya existe'
    });
  }
```

Base de Datos (MySQL):

```
-- Restricciones de la tabla
CREATE TABLE Categoria (
  idCategoria INT(11) NOT NULL AUTO_INCREMENT,
  nombre VARCHAR(50) NOT NULL UNIQUE,  -- ← Evita duplicados
  descripcion VARCHAR(255),
  estado TINYINT(1) DEFAULT 1,
  PRIMARY KEY (idCategoria)
);
```

🎯 ARCHIVOS INVOLUCRADOS

Frontend:

1. `src/pages/Dashboard.jsx` (líneas 1-3741)
   - Renderiza tabla de categorías
   - Botón "Nueva Categoría"
   - Maneja estado `categoryModalOpen`
2. `src/components/CategoryModal.jsx` (completo)
   - Formulario de creación
   - Validación frontend
   - Llamada a API
3. `src/utils/api.js` (líneas 1-50)
   - Configuración Axios
   - Interceptor para JWT
   - Manejo de errores
4. `src/components/ToastProvider.jsx` (completo)
   - Notificaciones de éxito/error
   - Animaciones

Backend:

5. `index.js` (líneas 300-400 aprox.)
    - Route: `POST /api/categorias`
    - Middleware: `verifyToken`
    - Validación con `express-validator`
    - Lógica de inserción
6. `db.js` (completo)
    - Conexión MySQL
    - Pool de conexiones

Base de Datos:

7. `ESTRUCTURA_BD_COMPLETA.sql` (tabla Categoria)
    - Definición de estructura
    - Índices y restricciones

MANEJO DE ERRORES

Posibles Errores:

Error	Causa	Mensaje	Código HTTP
Token inválido	JWT expirado o manipulado	"Token inválido o expirado"	401
Nombre vacío	Validación frontend/backend	"El nombre es requerido"	400
Nombre duplicado	Ya existe en BD	"La categoría ya existe"	400
Error de BD	MySQL no disponible	"Error al crear categoría"	500
Sin conexión	Red no disponible	"Error de conexión"	-

Código de Manejo:

```
// En CategoryModal.jsx
try {
  const response = await api.post('/api/categorias', formData);
  if (response.data.success) {
    // Éxito
    showToast('Categoría creada exitosamente', 'success');
    onCategoriaCreated(response.data.data);
    onClose();
  }
} catch (error) {
  if (error.response) {
    // Error del servidor (4xx, 5xx)
    showToast(error.response.data.message || 'Error al crear', 'error');
  } else if (error.request) {
    // No hay respuesta del servidor
```

```
      showToast('Sin conexión al servidor', 'error');
    } else {
      // Error en la configuración
      showToast('Error inesperado', 'error');
    }
  }
```

MEJORAS Y OPTIMIZACIONES

Implementadas:

- ☑ Validación en múltiples capas (frontend + backend + BD)
- ☑ Feedback inmediato al usuario (toast notifications)
- ☑ Actualización optimista de la UI
- ☑ Manejo robusto de errores
- ☑ JWT para seguridad
- ☑ Índices en BD para rendimiento

Posibles Mejoras Futuras:

- Agregar debounce a la validación de nombre duplicado
- Permitir subir imagen de la categoría
- Agregar tags o etiquetas a categorías
- Mostrar contador de productos por categoría
- Búsqueda en tiempo real de categorías existentes
- Cache de categorías en frontend (Redux/Context)

CONCLUSIÓN

Este diagrama muestra el flujo completo de **7 pasos** para crear una categoría:

1. ☑ Usuario inicia proceso desde Dashboard
2. ☑ Completa formulario en CategoryModal
3. ☑ Envío seguro con JWT al backend
4. ☑ Validación exhaustiva en backend
5. ☑ Inserción en base de datos MySQL
6. ☑ Respuesta estructurada al frontend
7. ☑ Actualización inmediata de la UI

Total de validaciones: 6 niveles
Total de archivos involucrados: 7
Tiempo de respuesta: < 2 segundos
Seguridad: JWT + Express Validator + UNIQUE constraint

REFERENCIAS

- Código fuente: `src/components/CategoryModal.jsx`
- API endpoint: `POST /api/categorias` (línea ~350 de `index.js`)
- Tabla BD: `Categoria` en `ESTRUCTURA_BD_COMPLETA.sql`
- Documentación completa: Ver `DOCUMENTACION/04-COMPONENTES-FRONTEND.md`

Fecha de creación: 23 de octubre de 2025
Autor: Sistema de Documentación Automática
Versión: 1.0 - Diagrama Mejorado Sin Cruces