🔧 Documentación Técnica - Sistema Empresarial

📋 Tabla de Contenidos

🏗️ Arquitectura de Componentes

Estructura Jerárquica

```
src/
├── 📁 components/
│   ├── 📁 ui/                    # Componentes base reutilizables
│   │   ├── button.jsx            # Botón base con variantes
│   │   ├── card.jsx              # Contenedores de información
│   │   ├── dialog.jsx            # Modales y diálogos
│   │   ├── form.jsx              # Elementos de formulario
│   │   ├── input.jsx             # Campos de entrada
│   │   ├── table.jsx             # Tablas de datos
│   │   ├── toast.jsx             # Notificaciones
│   │   ├── badge.jsx             # Etiquetas y badges
│   │   ├── progress.jsx          # Barras de progreso
│   │   ├── select.jsx            # Selectores dropdown
│   │   ├── tabs.jsx              # Pestañas
│   │   ├── alert-dialog.jsx      # Confirmaciones
│   │   ├── import-dialog.jsx     # Dialog de importación
│   │   └── template-downloader.jsx # Descarga de plantillas
│   │
│   ├── 📁 forms/                 # Componentes de formularios específicos
│   │   ├── FacturaForm.jsx       # Formulario de facturas
│   │   ├── TransaccionForm.jsx   # Formulario de transacciones
│   │   ├── TerceroForm.jsx       # Formulario de terceros
│   │   ├── ContratoForm.jsx      # Formulario de contratos
│   │   └── ImpuestoForm.jsx      # Formulario de impuestos
│   │
│   └── 📁 layout/                # Componentes de layout
│       ├── Header.jsx            # Cabecera principal
│       ├── Sidebar.jsx           # Menú lateral
│       ├── Footer.jsx            # Pie de página
│       └── MainLayout.jsx        # Layout principal
```

Patrón de Componentes Compuestos

Los componentes siguen el patrón de composición con subcomponentes:

```
// Ejemplo: Card Component
export const Card = ({ className, ...props }) => (
  <div className={cn("rounded-lg border bg-card text-card-foreground shadow-sm", className)}
{...props} />
);

Card.Header = ({ className, ...props }) => (
  <div className={cn("flex flex-col space-y-1.5 p-6", className)} {...props} />
);

Card.Title = ({ className, ...props }) => (
  <h3 className={cn("text-2xl font-semibold leading-none tracking-tight", className)}
{...props} />
);

Card.Content = ({ className, ...props }) => (
  <div className={cn("p-6 pt-0", className)} {...props} />
);
```

🎯 Patrones de Diseño

1. Container/Presentational Pattern

Containers: Manejan lógica y estado

```
// FacturasContainer.jsx
const FacturasContainer = () => {
  const [facturas, setFacturas] = useState([]);
  const [loading, setLoading] = useState(true);

  const fetchFacturas = async () => {
    try {
      const response = await api.get('/facturas');
      setFacturas(response.data);
    } catch (error) {
      toast.error('Error cargando facturas');
    } finally {
      setLoading(false);
    }
  };

  return (
    <FacturasList
      facturas={facturas}
      loading={loading}
      onRefresh={fetchFacturas}
    />
  );
};
```

Presentational: Solo renderizan UI

```
// FacturasList.jsx
const FacturasList = ({ facturas, loading, onRefresh }) => {
  if (loading) return <Skeleton />;

  return (
    <div className="space-y-4">
      {facturas.map(factura => (
        <FacturaCard key={factura.id} factura={factura} />
      ))}
    </div>
  );
};
```

2. Custom Hooks Pattern

```
// hooks/useFacturas.js
export const useFacturas = () => {
  const [facturas, setFacturas] = useState([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);

  const fetchFacturas = useCallback(async () => {
    try {
      setLoading(true);
      const response = await api.get('/api/facturas');
      setFacturas(response.data);
    } catch (err) {
      setError(err.message);
    } finally {
      setLoading(false);
    }
  }, []);

  const createFactura = useCallback(async (data) => {
    try {
      const response = await api.post('/api/facturas', data);
      setFacturas(prev => [...prev, response.data]);
      return response.data;
    } catch (err) {
      throw err;
    }
  }, []);

  useEffect(() => {
    fetchFacturas();
  }, [fetchFacturas]);

  return {
```

```
    facturas,
    loading,
    error,
    fetchFacturas,
    createFactura
  };
};
```

🗄️ Estructura de Base de Datos

Modelos Principales

Facturas

```
{
  id_factura: INTEGER (PK),
  numero_factura: STRING,
  estatus_factura: ENUM ['PENDIENTE', 'PAGADA', 'VENCIDA'],
  fecha_radicado: DATE,
  fecha_estimada_pago: DATE,
  subtotal_facturado_moneda: DECIMAL,
  id_contrato: INTEGER (FK),
  id_moneda: INTEGER (FK)
}
```

Transacciones

```
{
  id_transaccion: INTEGER (PK),
  fecha_transaccion: DATE,
  valor_total_transaccion: DECIMAL,
  tipo_transaccion: ENUM ['VENTA', 'COMPRA', 'PAGO', 'REEMBOLSO'],
  descripcion_transaccion: TEXT,
  id_cuenta: INTEGER (FK),
  id_tipo_transaccion: INTEGER (FK),
  id_moneda: INTEGER (FK)
}
```

Terceros

```
{
  id_tercero: INTEGER (PK),
  nombre_tercero: STRING,
  tipo_personalidad: ENUM ['NATURAL', 'JURIDICA'],
  numero_identificacion: STRING,
  estado_tercero: ENUM ['ACTIVO', 'INACTIVO'],
  telefono_tercero: STRING,
  email_tercero: STRING,
```

```
  direccion_tercero: TEXT
}
```

🔗 APIs y Controladores

Endpoints Principales

Facturas

- `GET /api/facturas` - Listar facturas con paginación
- `GET /api/facturas/:id` - Obtener factura específica
- `POST /api/facturas` - Crear nueva factura
- `PUT /api/facturas/:id` - Actualizar factura
- `DELETE /api/facturas/:id` - Eliminar factura
- `POST /api/facturas/import` - Importar desde Excel

Transacciones

- `GET /api/transacciones` - Listar transacciones
- `GET /api/transacciones/:id` - Obtener transacción
- `POST /api/transacciones` - Crear transacción

Terceros

- `GET /api/terceros` - Listar terceros
- `POST /api/terceros` - Crear tercero
- `POST /api/terceros/bulk-create` - Creación masiva

Estructura de Respuestas

```
// Respuesta exitosa con paginación
{
  "data": [...],
  "pagination": {
    "page": 1,
    "limit": 10,
    "total": 100,
    "pages": 10
  }
}

// Respuesta de error
{
  "error": "Descripción del error",
  "details": [...] // Opcional: detalles adicionales
}
```

👨‍💻 Guía de Desarrollo

Configuración del Entorno

1. **Clonar repositorio**

```
git clone <repo-url>
cd sistema-empresarial
```

2. **Instalar dependencias**

```
npm install
```

3. **Configurar base de datos**

```
# Crear base de datos PostgreSQL
createdb SQL_DDL_ADMCOT

# Configurar variables en src/config/database.js
```

4. **Ejecutar aplicación**

```
# Terminal 1: Backend
npm run server

# Terminal 2: Frontend
npm run dev
```

Flujo de Desarrollo

1. **Crear nueva feature**

```
git checkout -b feature/nueva-funcionalidad
```

2. **Desarrollo**
   - Crear/modificar componentes en `src/components/`
   - Crear/modificar páginas en `src/pages/`
   - Crear/modificar APIs en `src/routes/` y `src/controllers/`
   - Actualizar modelos en `src/models/` si es necesario

3. **Testing**

```
npm run test
```

4. **Commit y push**

```
git add .
git commit -m "feat: descripción de la funcionalidad"
git push origin feature/nueva-funcionalidad
```

📏 Estándares de Código

Convenciones de Nombres

- **Componentes**: PascalCase (`FacturaForm.jsx`)
- **Funciones**: camelCase (`fetchFacturas`)
- **Constantes**: UPPER_SNAKE_CASE (`ESTADOS_FACTURA`)
- **Archivos**: kebab-case para utilidades, PascalCase para componentes
- **Variables de BD**: snake_case (`id_factura`)

Estructura de Archivos

```
// Importaciones (orden específico)
import React, { useState, useEffect } from 'react'; // React primero
import { useNavigate } from 'react-router-dom';      // Librerías externas
import { Button } from '@/components/ui/button';     // Componentes internos
import { useFacturas } from '@/hooks/useFacturas';   // Hooks custom
import './Component.css';                            // Estilos (si aplica)

// Componente
const MiComponente = () => {
  // 1. Estados
  const [loading, setLoading] = useState(false);

  // 2. Hooks
  const navigate = useNavigate();
  const { facturas, fetchFacturas } = useFacturas();

  // 3. Effects
  useEffect(() => {
    fetchFacturas();
  }, []);

  // 4. Handlers
  const handleSubmit = async (data) => {
    // lógica
  };

  // 5. Render
  return (
    <div>
      {/* JSX */}
    </div>
  );
};

export default MiComponente;
```

Validaciones

Usar React Hook Form para formularios:

```jsx
import { useForm } from 'react-hook-form';

const MiFormulario = () => {
  const {
    register,
    handleSubmit,
    formState: { errors },
    reset
  } = useForm();

  const onSubmit = async (data) => {
    try {
      await api.post('/endpoint', data);
      toast.success('Guardado exitosamente');
      reset();
    } catch (error) {
      toast.error('Error al guardar');
    }
  };

  return (
    <form onSubmit={handleSubmit(onSubmit)}>
      <input
        {...register('nombre', {
          required: 'Nombre es requerido',
          minLength: { value: 2, message: 'Mínimo 2 caracteres' }
        })}
        placeholder="Nombre"
      />
      {errors.nombre && <span>{errors.nombre.message}</span>}

      <button type="submit">Guardar</button>
    </form>
  );
};
```

🧪 Testing

Configurar Vitest

```jsx
// vite.config.js
export default {
  test: {
    environment: 'jsdom',
    setupFiles: ['./src/test/setup.js'],
  },
}
```

Ejemplos de Tests

```jsx
// __tests__/components/Button.test.jsx
import { render, screen, fireEvent } from '@testing-library/react';
import { describe, it, expect, vi } from 'vitest';
import { Button } from '@/components/ui/button';

describe('Button', () => {
  it('renderiza correctamente', () => {
    render(<Button>Click me</Button>);
    expect(screen.getByText('Click me')).toBeInTheDocument();
  });

  it('ejecuta onClick cuando se hace clic', () => {
    const handleClick = vi.fn();
    render(<Button onClick={handleClick}>Click me</Button>);

    fireEvent.click(screen.getByText('Click me'));
    expect(handleClick).toHaveBeenCalledTimes(1);
  });
});
```

🚀 Deployment

Build de Producción

```bash
# Construir frontend
npm run build

# El output estará en dist/
```

Variables de Entorno Producción

```
NODE_ENV=production
DATABASE_URL=postgresql://user:pass@host:port/dbname
PORT=5000
CORS_ORIGIN=https://tu-dominio.com
```

Nginx Configuration

```nginx
server {
    listen 80;
    server_name tu-dominio.com;

    # Frontend estático
    location / {
        root /var/www/sistema-empresarial/dist;
        try_files $uri $uri/ /index.html;
    }
```

```
    # API Backend
    location /api {
        proxy_pass http://localhost:5000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
}
```

📚 Recursos Adicionales

- Guía de React Hooks
- Documentación de Sequelize
- Tailwind CSS Cheatsheet
- Testing Library

Documentación técnica v1.0 - Actualizada: Junio 2025