Somos Crédito

Módulo de Gestión de Sucursales
(Manual Técnico)

Oscar Flores
27 de Noviembre del 2025

1. Introducción

1.1 Descripción general del sistema

El sistema desarrollado permite la **gestión de sucursales** de la institución *Somos Crédito*, ofreciendo un conjunto de funcionalidades que facilitan el mantenimiento de la información de cada sucursal.

Las principales funciones implementadas son:

- **CRUD de sucursales**: creación, lectura, actualización y eliminación de registros.
- **Filtros de búsqueda** por nombre, municipio, departamento y estado.
- **Paginación** de resultados para una mejor experiencia de usuario.
- **Exportación a PDF** del listado de sucursales utilizando jsPDF y jspdf-autotable.
- Manejo de **alertas de confirmación y éxito/error** con **SweetAlert2**.
- Resumen visual con **total de sucursales, activas e inactivas**.

1.2 Tecnologías utilizadas

El proyecto está dividido en dos partes: **backend (API REST)** y **frontend (SPA en React)**.

- **Backend**
    - **Node.js + Express**
    - **Sequelize** como ORM
    - **MySQL** como base de datos
    - Manejo de CORS para permitir el consumo desde el frontend

- **Frontend**
    - **React** utilizando **Hooks** (cumpliendo el requisito del enunciado).
    - **Axios** para las peticiones HTTP a la API.
    - **SweetAlert2** para alertas de confirmación y mensajes al usuario.
    - **jsPDF** + **jspdf-autotable** para generación de reportes en PDF.

2. Requisitos

2.1 Software necesario

Para ejecutar correctamente el sistema se requiere:

- **Node.js** versión recomendada: **18.x o superior** (compatible con Express 5 y Vite 7).
- **NPM** (incluido con Node.js).
- **MySQL** (versión 5.7 o 8.x).
- Editor de código recomendado: **Visual Studio Code**.
- Navegador moderno (Chrome, Edge, Brave, Firefox).

2.2 Dependencias principales del backend

El backend utiliza Node.js con Express y Sequelize como ORM. Estas son las dependencias reales del archivo package.json:

Backend (somos-credito-backend)

Paquete	Versión	Descripción
express	^5.1.0	Framework para manejo de rutas HTTP y servidor web.
sequelize	^6.37.7	ORM para interactuar con MySQL usando modelos y consultas estructuradas.
mysql2	^3.15.3	Driver oficial para conectarse a MySQL.
cors	^2.8.5	Permite al frontend consumir la API desde otro origen.

2.3 Dependencias principales del frontend

El frontend está construido con React 19, Vite y Axios.

Frontend (somos-credito-frontend)

Paquete	Versión	Descripción
react	^19.2.0	Librería principal para la interfaz gráfica.
react-dom	^19.2.0	Renderizado de componentes React en el DOM.
axios	^1.13.2	Cliente HTTP para consumir la API backend.
sweetalert2	^11.26.3	Alertas estilizadas para confirmaciones y mensajes al usuario.
jspdf	^3.0.4	Biblioteca para generar documentos PDF.
jspdf-autotable	^5.0.2	Extensión para crear tablas dinámicas dentro del PDF.

Dependencias de desarrollo (devDependencies)

Paquete	Versión	Función
vite	^7.2.4	Herramienta moderna de desarrollo para React.
@vitejs/plugin-react	^5.1.1	Plugin necesario para usar React con Vite.
eslint	^9.39.1	Linter para mantener el código limpio.
eslint-plugin-react-hooks	^7.0.1	Reglas especiales para garantizar el correcto uso de Hooks.
eslint-plugin-react-refresh	^0.4.24	Soporte para React Fast Refresh.
@types/react	^19.2.5	Tipados para React (TypeScript).
@types/react-dom	^19.2.3	Tipados para React DOM.
globals	^16.5.0	Tipos globales para ESLint.

3. Arquitectura general

3.1 Descripción de la arquitectura

El sistema implementa una arquitectura basada en **tres capas bien definidas**, lo cual facilita la escalabilidad, el mantenimiento y la separación de responsabilidades.

1. **Capa de Presentación (Frontend – React)**

    - Desarrollada como una SPA basada en React 19.
    - Utiliza Hooks para el manejo de estado y lógica interna.
    - Realiza llamadas HTTP mediante Axios hacia el backend.
    - Presenta la información en tablas, formularios y filtros.
      Permite exportación de datos a PDF mediante jsPDF + AutoTable.

2. **Capa de Lógica de Negocio (Backend – Node.js + Express)**

    - Expone servicios RESTful bajo /api/sucursales.
    - Implementa middleware para CORS, parsing JSON y manejo de errores.
    - Contiene controladores que procesan las peticiones y validan los datos.
    - Usa Sequelize como ORM para evitar consultas SQL manuales.
    - Gestiona las operaciones CRUD sobre el modelo Sucursal.

3. **Capa de Datos (MySQL)**

    - Base de datos relacional donde se almacena la información de sucursales.
    - Incluye campos como nombre, dirección, estado y fechas de auditoría.
    - Se interconecta con el backend mediante el driver mysql2 y Sequelize.

Gracias a esta arquitectura, cada parte del sistema puede evolucionar de manera independiente sin afectar las otras capas.

4. Base de datos

4.1 Estructura de la tabla sucursales

La aplicación utiliza una tabla principal llamada **sucursales**, donde se almacena toda la información relacionada con cada sucursal registrada.

A continuación se presenta el **script real utilizado** para crear la tabla:

```sql
CREATE TABLE `sucursales` (
  `id` int(11) NOT NULL,
  `nombre` varchar(100) NOT NULL,
  `calle_numero` varchar(100) NOT NULL,
  `colonia` varchar(100) DEFAULT NULL,
  `municipio` varchar(50) NOT NULL,
  `departamento` varchar(50) NOT NULL,
  `codigo_postal` varchar(10) DEFAULT NULL,
  `telefono` varchar(30) NOT NULL,
  `estado` enum('ACTIVA','INACTIVA','PENDIENTE') NOT NULL DEFAULT 'ACTIVA',
  `fecha_creacion` timestamp NOT NULL DEFAULT current_timestamp(),
  `fecha_modificacion` timestamp NOT NULL DEFAULT current_timestamp() ON UPDATE current_timestamp()
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

4.2 Descripción de los campos

Campo	Tipo / Longitud	Descripción
id	int(11) (PK)	Identificador único de la sucursal. Debe incrementarse automáticamente.
nombre	varchar(100)	Nombre comercial de la sucursal.
calle_numero	varchar(100)	Calle y número donde se ubica la sucursal.
colonia	varchar(100)	Colonia o sector (opcional).
municipio	varchar(50)	Municipio donde está ubicada la sucursal.
departamento	varchar(50)	Departamento de la sucursal.
codigo_postal	varchar(10)	Código postal (opcional).
telefono	varchar(30)	Número de teléfono de contacto.
estado	enum(...)	Estado de la sucursal: ACTIVA, INACTIVA o PENDIENTE. Por defecto: ACTIVA.
fecha_creacion	timestamp	Fecha en que el registro fue creado.
fecha_modificacion	timestamp	Fecha de última modificación. Se actualiza automáticamente.

4.3 Relación con el backend (Sequelize)

En el backend, esta tabla se modela mediante Sequelize, lo que permite:

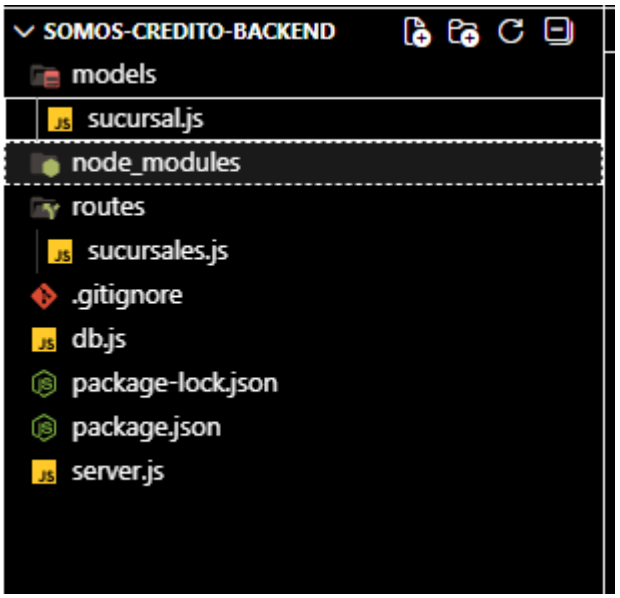
- Mapear cada campo con validaciones.
- Administrar registros mediante métodos del ORM.
- Sincronizar estructura si fuera necesario.

```
const { DataTypes } = require("sequelize");
const sequelize = require("../db");

const Sucursal = sequelize.define(
  "Sucursal",
  {
    id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true },
    nombre: { type: DataTypes.STRING(100), allowNull: false },
    calle_numero: { type: DataTypes.STRING(100), allowNull: false },
    colonia: { type: DataTypes.STRING(100), allowNull: true },
    municipio: { type: DataTypes.STRING(50), allowNull: false },
    departamento: { type: DataTypes.STRING(50), allowNull: false },
    codigo_postal: { type: DataTypes.STRING(10), allowNull: true },
    telefono: { type: DataTypes.STRING(30), allowNull: false },
    estado: {
      type: DataTypes.ENUM("ACTIVA", "INACTIVA", "PENDIENTE"),
      allowNull: false,
      defaultValue: "ACTIVA",
    },
    fecha_creacion: { type: DataTypes.DATE, allowNull: true },
    fecha_modificacion: { type: DataTypes.DATE, allowNull: true },
  },
  {
    tableName: "sucursales",
    timestamps: false,
  }
);

module.exports = Sucursal;
```

5. Backend (API REST)

5.1 Estructura de carpetas

La estructura básica del backend es:



5.2 Descripción de los archivos principales

- **server.js**
  Configura la aplicación Express, habilita CORS, parseo de JSON y monta las rutas de la API.
  También inicializa la conexión a la base de datos y levanta el servidor en el puerto configurado.

- **db.js**
  Contiene la configuración de Sequelize, incluyendo nombre de la base de datos, usuario, contraseña y host.
  Exporta la instancia de Sequelize utilizada por los modelos.

- **models/sucursal.js**
  Define el modelo Sucursal de Sequelize, mapeando la tabla sucursales y sus campos.
  Aquí se establecen tipos de datos, restricciones y el nombre de la tabla.

- **routes/sucursales.js**
  Define las rutas de la API relacionadas con las sucursales (GET, POST, PUT, DELETE).
  Cada ruta invoca las operaciones correspondientes sobre el modelo Sucursal.

5.3 Endpoints disponibles

La API expone los siguientes endpoints:

Método	Endpoint	Descripción
GET	/api/sucursales	Lista todas las sucursales (ordenadas desc).
POST	/api/sucursales	Crea una nueva sucursal.
PUT	/api/sucursales/:id	Actualiza los datos de una sucursal existente.
DELETE	/api/sucursales/:id	Elimina (o da de baja) una sucursal.

```
// GET /api/sucursales
router.get("/", async (req, res) => {
  try {
    const sucursales = await Sucursal.findAll({
      order: [["id", "DESC"]],
    });
    res.json(sucursales);
  } catch (err) {
    console.error("Error al listar sucursales:", err);
    res.status(500).json({ error: "Error al obtener sucursales" });
  }
});
```

```
// POST /api/sucursales
router.post("/", async (req, res) => {
  try {
    const nueva = await Sucursal.create({
      nombre: req.body.nombre,
      calle_numero: req.body.calle_numero,
      colonia: req.body.colonia || null,
      municipio: req.body.municipio,
      departamento: req.body.departamento,
      codigo_postal: req.body.codigo_postal || null,
      telefono: req.body.telefono,
      estado: req.body.estado || "ACTIVA",
    });
    res.json(nueva);
  } catch (err) {
    console.error("Error al crear sucursal:", err);
    res.status(400).json({ error: "Error al crear sucursal" });
  }
});
```

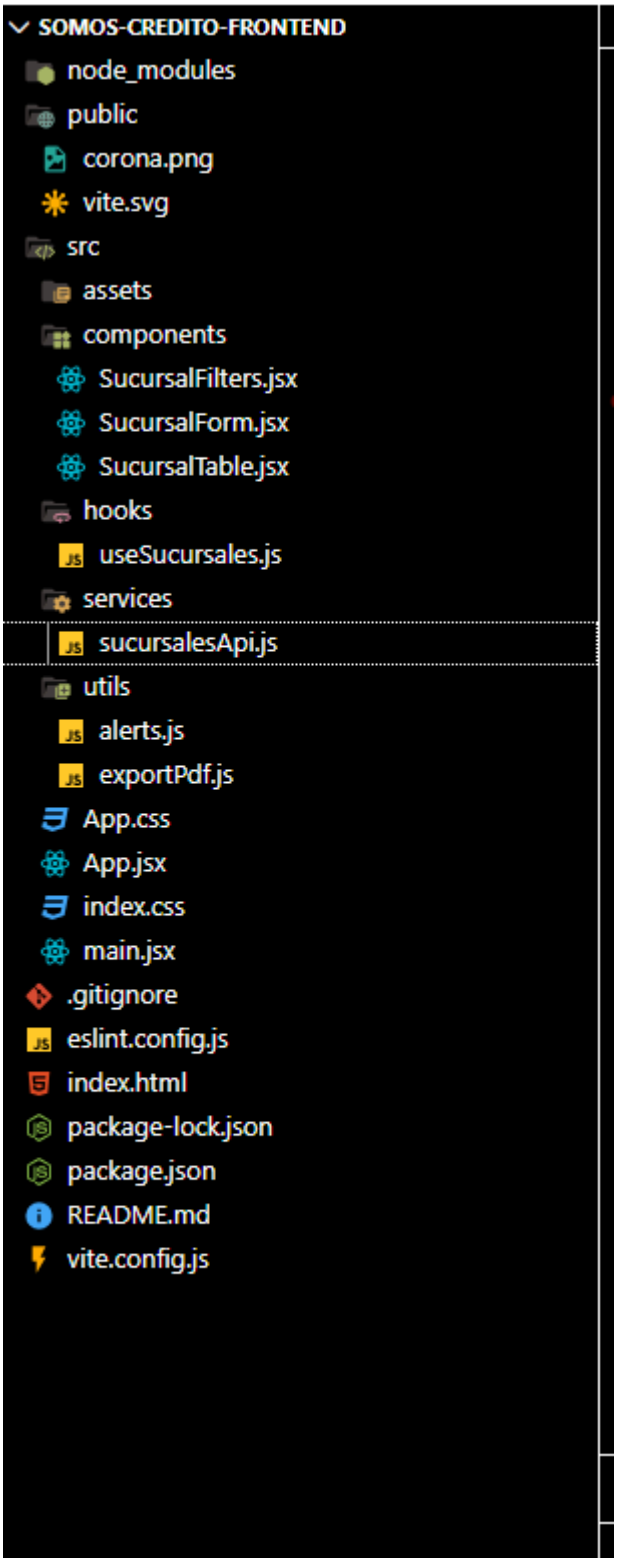
```
// PUT /api/sucursales/:id
router.put("/:id", async (req, res) => {
  try {
    const id = req.params.id;

    const [filasAfectadas] = await Sucursal.update(
      {
        nombre: req.body.nombre,
        calle_numero: req.body.calle_numero,
        colonia: req.body.colonia || null,
        municipio: req.body.municipio,
        departamento: req.body.departamento,
        codigo_postal: req.body.codigo_postal || null,
        telefono: req.body.telefono,
        estado: req.body.estado,
      },
      { where: { id } }
    );

    if (filasAfectadas === 0) {
      return res.status(404).json({ error: "Sucursal no encontrada" });
    }

    res.json({ mensaje: "Sucursal actualizada" });
  } catch (err) {
    console.error("Error al actualizar sucursal:", err);
    res.status(400).json({ error: "Error al actualizar sucursal" });
  }
});
```

```
// DELETE /api/sucursales/:id
router.delete("/:id", async (req, res) => {
  try {
    const id = req.params.id;
    const filas = await Sucursal.destroy({ where: { id } });

    if (filas === 0) {
      return res.status(404).json({ error: "Sucursal no encontrada" });
    }

    res.json({ mensaje: "Sucursal eliminada" });
  } catch (err) {
    console.error("Error al eliminar sucursal:", err);
    res.status(400).json({ error: "Error al eliminar sucursal" });
  }
});
```

6. Frontend

6.1 Estructura de carpetas

En el frontend se utilizó React con una organización por componentes, hooks y utilidades:

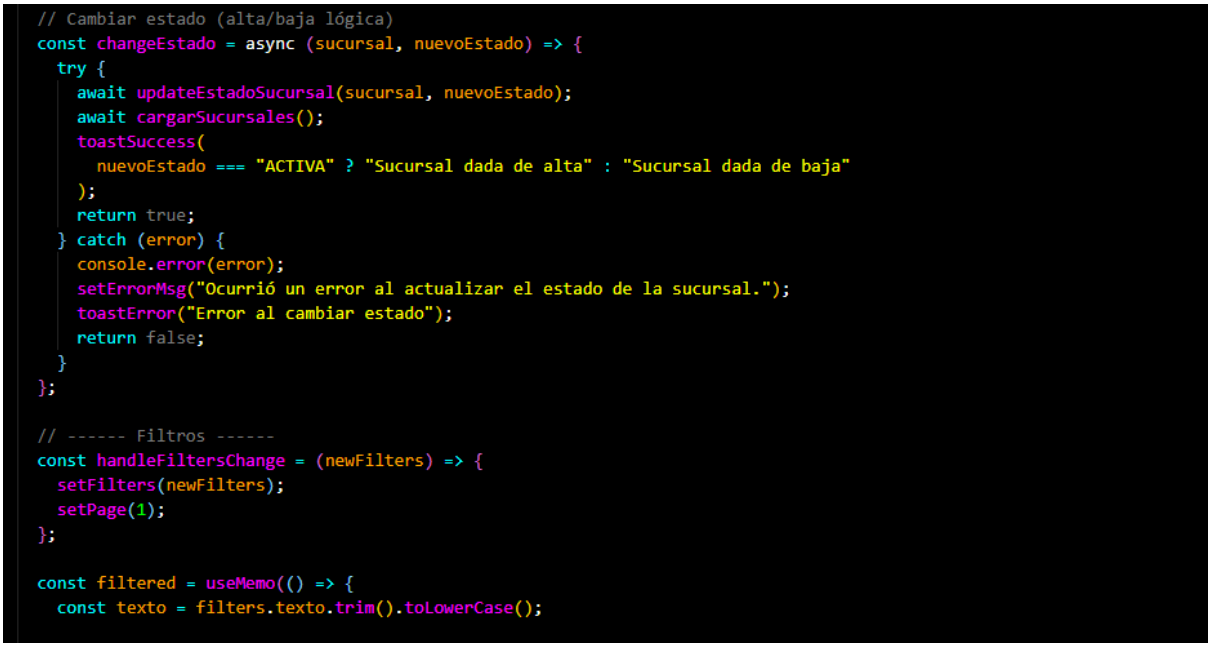


6.2 Hook useSucursales

El hook personalizado **useSucursales** centraliza la lógica de negocio del frontend:

- Maneja el **estado** de:
    - Lista de sucursales.
    - Filtros aplicados.
    - Página actual y tamaño de página.
    - Contadores (total, activas, inactivas).
- Se encarga de:
    - Llamar a la API a través de sucursalesApi.js.
    - Actualizar la tabla cuando cambian filtros o paginación.
    - Notificar errores y éxitos con SweetAlert2.

Esto cumple con el requisito del enunciado de utilizar **React Hooks** para el manejo de estado y flujo de datos.



```
// Cambiar estado (alta/baja lógica)
const changeEstado = async (sucursal, nuevoEstado) => {
  try {
    await updateEstadoSucursal(sucursal, nuevoEstado);
    await cargarSucursales();
    toastSuccess(
      nuevoEstado === "ACTIVA" ? "Sucursal dada de alta" : "Sucursal dada de baja"
    );
    return true;
  } catch (error) {
    console.error(error);
    setErrorMsg("Ocurrió un error al actualizar el estado de la sucursal.");
    toastError("Error al cambiar estado");
    return false;
  }
};

// ------ Filtros ------
const handleFiltersChange = (newFilters) => {
  setFilters(newFilters);
  setPage(1);
};

const filtered = useMemo(() => {
  const texto = filters.texto.trim().toLowerCase();
```

6.3 Componentes principales

- **SucursalForm.jsx**
  Componente que muestra el **formulario de alta/edición** de sucursal. Utiliza los estados provistos por useSucursales para:
    - Cargar datos cuando se edita una sucursal.
    - Limpiar el formulario después de guardar.
    - Validar los campos requeridos.

```jsx
function SucursalForm({ form, modoEdicion, onChange, onSubmit, onCancel }) {
  return (
    <div className="card">
      <h2 className="card-title">
        {modoEdicion ? "Editar sucursal" : "Registrar nueva sucursal"}
      </h2>

      <form onSubmit={onSubmit}>
        <div className="form-grid">

          <div className="form-group">
            <label>Nombre de la sucursal</label>
            <input
              type="text"
              name="nombre"
              placeholder="Ej: Sucursal Central Zona 1"
              value={form.nombre}
              onChange={onChange}
              required
            />
          </div>

          <div className="form-group">
            <label>Calle y número</label>
            <input
              type="text"
              name="calle_numero"
              placeholder="Ej: 6a Calle 3-15"
              value={form.calle_numero}
              onChange={onChange}
              required
            />
          </div>
```

- **SucursalTable.jsx**
  Muestra el **listado de sucursales** en una tabla, incluyendo:
  - Botones de editar y eliminar.
  - Paginación (botones de siguiente, anterior, etc.).
    Resumen o “badge” con:
    - Total de sucursales.
    - Activas.
    - Inactivas.

```jsx
function SucursalTable({
  sucursales,
  loading,
  page,
  rowsPerPage,
  totalRows,
  onPageChange,
  onEditar,
  onEliminar,
  onCambiarEstado,
}) {
  if (loading) return <p>Cargando...</p>;

  if (totalRows === 0) {
    return <p>No hay sucursales que coincidan con el filtro.</p>;
  }

  const totalPages = Math.max(1, Math.ceil(totalRows / rowsPerPage));

  return (
    <div className="card">
      <div className="table-wrapper">
        <table className="table">
          <thead>
            <tr>
              <th>ID</th>
              <th>Nombre</th>
              <th>Dirección</th>
              <th>Municipio / Depto.</th>
              <th>Teléfono</th>
              <th>Estado</th>
              <th>Acciones</th>
            </tr>
          </thead>
          <tbody>
```

- **SucursalFilters.jsx**
  Contiene los **campos de filtro**:

    - Búsqueda por nombre.
      Selección de municipio/departamento.
    - Filtro por estado (activo/inactivo).
    - Cada cambio en los filtros dispara una actualización de la lista a través del hook.

```jsx
function SucursalFilters({ filters, onChange }) {
  const handleChange = (e) => {
    onChange({
      ...filters,
      [e.target.name]: e.target.value,
    });
  };

  return (
    <div className="filters-bar">
      <input
        type="text"
        name="texto"
        placeholder="Buscar por nombre, municipio o departamento..."
        value={filters.texto}
        onChange={handleChange}
      />
      <select
        name="estado"
        value={filters.estado}
        onChange={handleChange}
      >
        <option value="TODOS">Todos los estados</option>
        <option value="ACTIVA">Activas</option>
        <option value="INACTIVA">Inactivas</option>
        <option value="PENDIENTE">Pendientes</option>
      </select>
    </div>
  );
}

export default SucursalFilters;
```

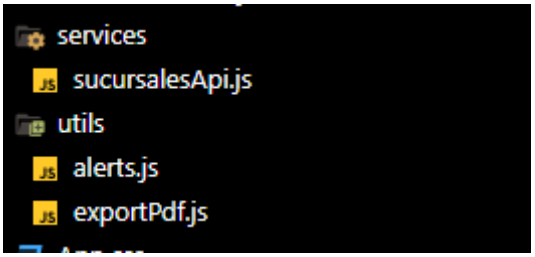
6.4 Servicios y utilidades

- **sucursalesApi.js**
  Encapsula las llamadas a la API con Axios (GET, POST, PUT, DELETE) para no repetir lógica en los componentes.

- **exportPDF.js**
  Define la función que recibe la lista de sucursales y genera un **PDF** usando jsPDF y jspdf-autotable.

- **alerts.js**
  Centraliza las funciones de SweetAlert2:

    - Confirmación al eliminar.
    - Mensajes de éxito y error.




```js
import axios from "axios";

const API_URL = "http://localhost:3001/api/sucursales";

export const getSucursales = () => {
  return axios.get(API_URL);
};

export const createSucursal = (data) => {
  return axios.post(API_URL, data);
};

export const updateSucursal = (id, data) => {
  return axios.put(`${API_URL}/${id}`, data);
};

export const deleteSucursal = (id) => {
  return axios.delete(`${API_URL}/${id}`);
};

// cambiar solo estado (alta / baja lógica)
export const updateEstadoSucursal = (sucursal, nuevoEstado) => {
  return axios.put(`${API_URL}/${sucursal.id}`, {
    ...sucursal,
    estado: nuevoEstado,
  });
};
```

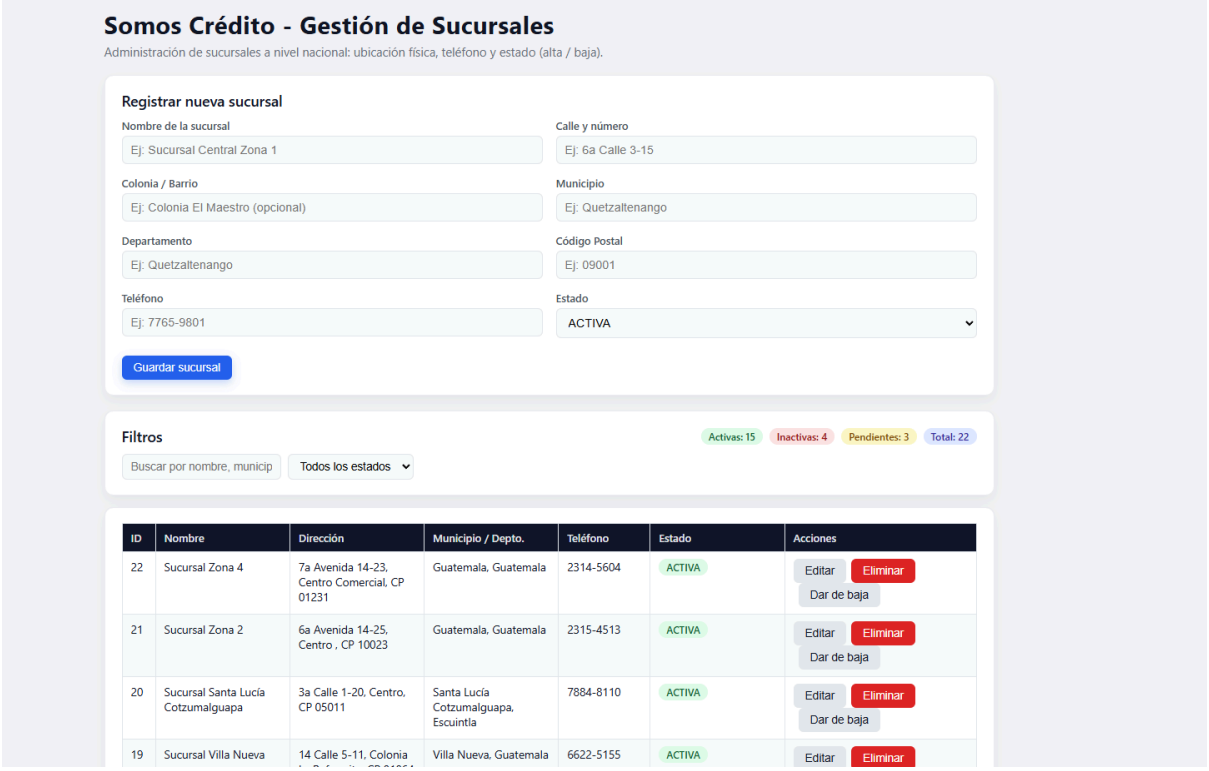
```js
import Swal from "sweetalert2";

export function toastSuccess(title = "Operación exitosa", text = "") {
  return Swal.fire({
    icon: "success",
    title,
    text,
    timer: 1800,
    showConfirmButton: false,
    position: "top-end",
    toast: true,
  });
}

export function toastError(title = "Ocurrió un error", text = "") {
  return Swal.fire({
    icon: "error",
    title,
    text,
    timer: 2200,
    showConfirmButton: false,
    position: "top-end",
    toast: true,
  });
}

export function confirmDelete(text = "Esta acción no se puede deshacer") {
  return Swal.fire({
    title: "¿Estás seguro?",
    text,
    icon: "warning",
    showCancelButton: true,
    confirmButtonColor: "#dc2626",
    cancelButtonColor: "#6b7280",
```

```js
export function exportSucursalesPdf(rows, titulo = "Listado de sucursales", filename = "sucursales.pdf") {
  const doc = new jsPDF();

  doc.setFontSize(14);
  doc.text(titulo, 14, 15);

  const head = [["ID", "Nombre", "Dirección", "Municipio / Depto.", "Teléfono", "Estado"]];
  const body = rows.map((s) => [
    s.id,
    s.nombre,
    `${s.calle_numero}${s.colonia ? ", " + s.colonia : ""}${s.codigo_postal ? ", CP " + s.codigo_postal : ""}`,
    `${s.municipio}, ${s.departamento}`,
    s.telefono,
    s.estado,
  ]);

  autoTable(doc, {
    head,
    body,
    startY: 22,
    styles: { fontSize: 8 },
  });

  doc.save(filename);
```

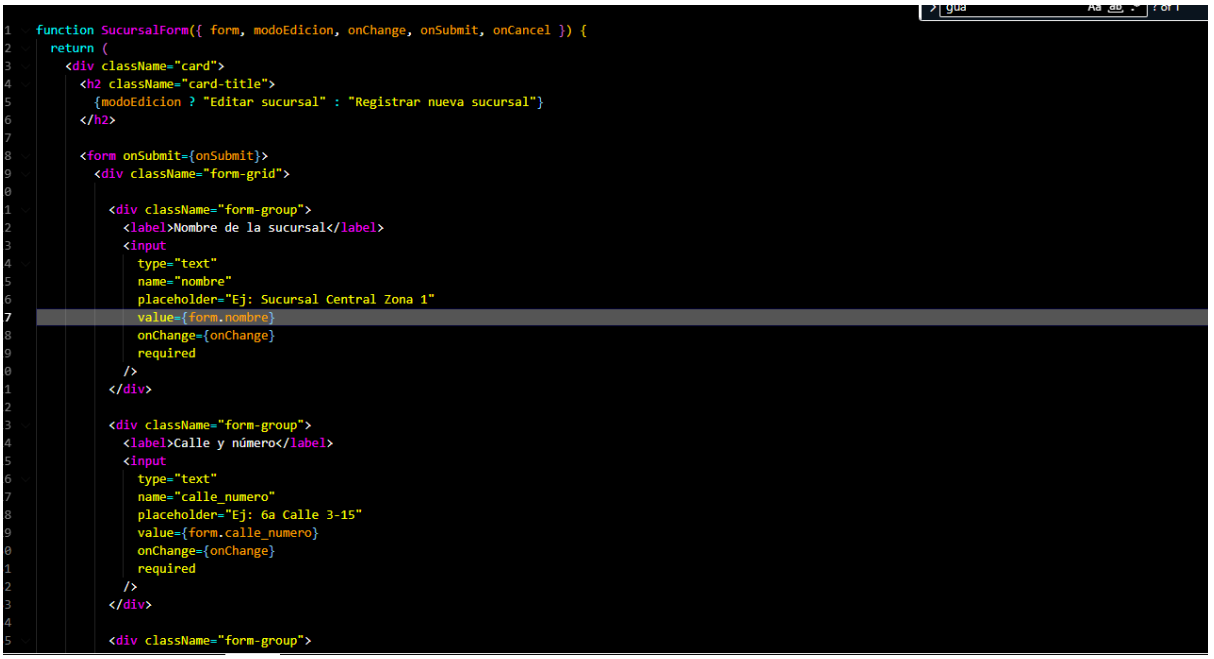
7. Capturas de la aplicación en ejecución

1. **Formulario vacío** de registro de sucursal.

Vista



Código



```
function SucursalForm({ form, modoEdicion, onChange, onSubmit, onCancel }) {
  return (
    <div className="card">
      <h2 className="card-title">
        {modoEdicion ? "Editar sucursal" : "Registrar nueva sucursal"}
      </h2>

      <form onSubmit={onSubmit}>
        <div className="form-grid">

          <div className="form-group">
            <label>Nombre de la sucursal</label>
            <input
              type="text"
              name="nombre"
              placeholder="Ej: Sucursal Central Zona 1"
              value={form.nombre}
              onChange={onChange}
              required
            />
          </div>

          <div className="form-group">
            <label>Calle y número</label>
            <input
              type="text"
              name="calle_numero"
              placeholder="Ej: 6a Calle 3-15"
              value={form.calle_numero}
              onChange={onChange}
              required
            />
          </div>

          <div className="form-group">
```

2. **Listado de sucursales** con paginación.

Vista

ID	Nombre	Dirección	Municipio / Depto.	Teléfono	Estado	Acciones
22	Sucursal Zona 4	7a Avenida 14-23, Centro Comercial, CP 01231	Guatemala, Guatemala	2314-5604	ACTIVA	Editar Eliminar Dar de baja
21	Sucursal Zona 2	6a Avenida 14-25, Centro , CP 10023	Guatemala, Guatemala	2315-4513	ACTIVA	Editar Eliminar Dar de baja
20	Sucursal Santa Lucía Cotzumalguapa	3a Calle 1-20, Centro, CP 05011	Santa Lucía Cotzumalguapa, Escuintla	7884-8110	ACTIVA	Editar Eliminar Dar de baja
19	Sucursal Villa Nueva	14 Calle 5-11, Colonia La Reformita, CP 01064	Villa Nueva, Guatemala	6622-5155	ACTIVA	Editar Eliminar Dar de baja
18	Sucursal Amatitlán	7a Avenida 5-80, Colonia Progreso, CP 01063	Amatitlán, Guatemala	6644-2215	ACTIVA	Editar Eliminar Dar de baja

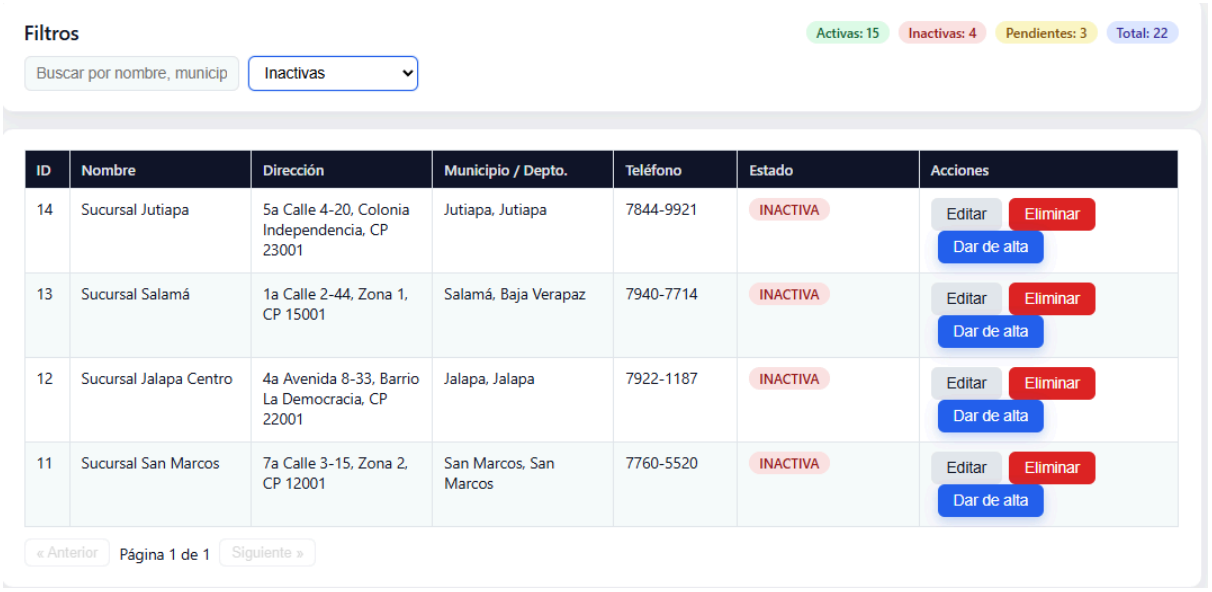
« Anterior Página 1 de 5 Siguiente »

Código

```
function SucursalTable({
  sucursales,
  loading,
  page,
  rowsPerPage,
  totalRows,
  onPageChange,
  onEditar,
  onEliminar,
  onCambiarEstado,
}) {
  if (loading) return <p>Cargando...</p>;

  if (totalRows === 0) {
    return <p>No hay sucursales que coincidan con el filtro.</p>;
  }

  const totalPages = Math.max(1, Math.ceil(totalRows / rowsPerPage));

  return (
    <div className="card">
      <div className="table-wrapper">
        <table className="table">
          <thead>
            <tr>
              <th>ID</th>
              <th>Nombre</th>
              <th>Dirección</th>
              <th>Municipio / Depto.</th>
              <th>Teléfono</th>
              <th>Estado</th>
              <th>Acciones</th>
            </tr>
          </thead>
```

3. **Filtros aplicados** (por ejemplo, solo sucursales activas de cierto municipio).

Vista



Código

```jsx
function SucursalFilters({ filters, onChange }) {
  const handleChange = (e) => {
    onChange({
      ...filters,
      [e.target.name]: e.target.value,
    });
  };

  return (
    <div className="filters-bar">
      <input
        type="text"
        name="texto"
        placeholder="Buscar por nombre, municipio o departamento..."
        value={filters.texto}
        onChange={handleChange}
      />
      <select
        name="estado"
        value={filters.estado}
        onChange={handleChange}
      >
        <option value="TODOS">Todos los estados</option>
        <option value="ACTIVA">Activas</option>
        <option value="INACTIVA">Inactivas</option>
        <option value="PENDIENTE">Pendientes</option>
      </select>
    </div>
  );
}

export default SucursalFilters;
```

4. **PDF generado** abierto en un visor (por ejemplo, la tabla exportada).

Vista

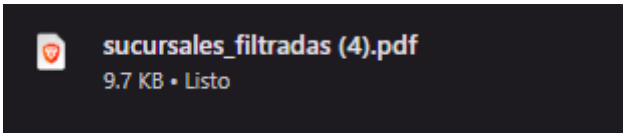
Sucursales - Resultados filtrados

ID	Nombre	Dirección	Municipio / Depto.	Teléfono	Estado
14	Sucursal Jutiapa	5a Calle 4-20, Colonia Independencia, CP 23001	Jutiapa, Jutiapa	7844-9921	INACTIVA
13	Sucursal Salamá	1a Calle 2-44, Zona 1, CP 15001	Salamá, Baja Verapaz	7940-7714	INACTIVA
12	Sucursal Jalapa Centro	4a Avenida 8-33, Barrio La Democracia, CP 22001	Jalapa, Jalapa	7922-1187	INACTIVA
11	Sucursal San Marcos	7a Calle 3-15, Zona 2, CP 12001	San Marcos, San Marcos	7760-5520	INACTIVA

Código

```
export function exportSucursalesPdf(rows, titulo = "Listado de sucursales", filename = "sucursales.pdf") {
  const doc = new jsPDF();

  doc.setFontSize(14);
  doc.text(titulo, 14, 15);

  const head = [["ID", "Nombre", "Dirección", "Municipio / Depto.", "Teléfono", "Estado"]];
  const body = rows.map((s) => [
    s.id,
    s.nombre,
    `${s.calle_numero}${s.colonia ? ", " + s.colonia : ""}${s.codigo_postal ? ", CP " + s.codigo_postal : ""}`,
    `${s.municipio}, ${s.departamento}`,
    s.telefono,
    s.estado,
  ]);

  autoTable(doc, {
    head,
    body,
    startY: 22,
    styles: { fontSize: 8 },
  });

  doc.save(filename);
}
```

Descarga

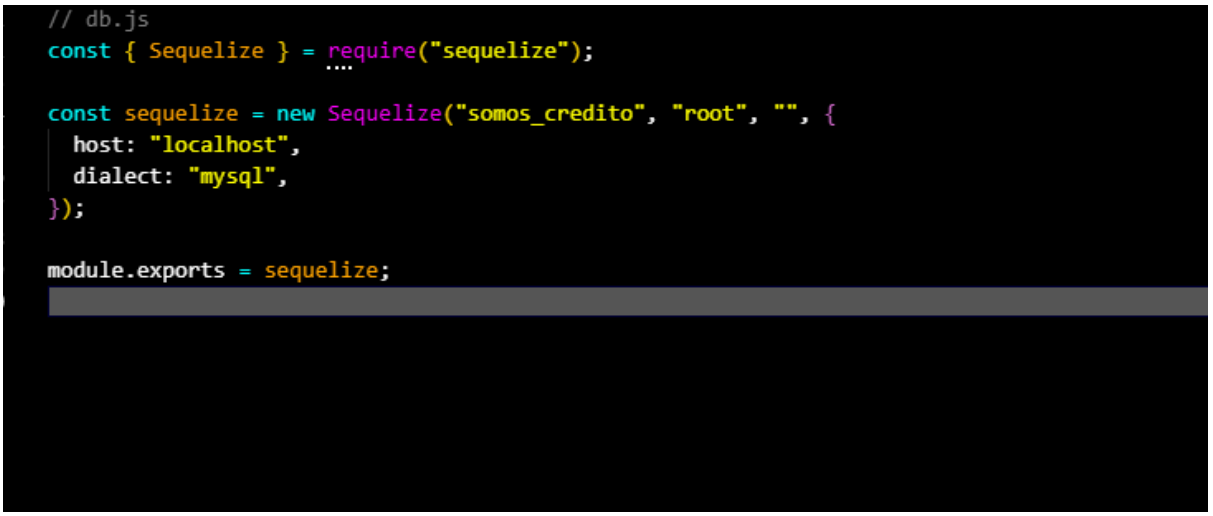


8. Instrucciones de despliegue / ejecución

En un entorno real, la configuración de la base de datos y URLs de la API se moverían a variables de entorno (.env), pero para simplificar la prueba técnica se dejó la configuración directa en los archivos db.js y sucursalesApi.js.

8.1 Backend

1. Clonar o descargar el proyecto backend: cd somos-credito-backend
2. Instalar dependencias: npm install
3. Configurar las variables de conexión a MySQL (por ejemplo en db.js o .env).
4. Ejecutar migraciones o crear la tabla sucursales si no existe.
5. Levantar el servidor: node server.js
6. El backend quedará escuchando en http://localhost:3000 (o el puerto configurado).



```
// db.js
const { Sequelize } = require("sequelize");

const sequelize = new Sequelize("somos_credito", "root", "", {
  host: "localhost",
  dialect: "mysql",
});

module.exports = sequelize;
```

8.2 Frontend

1. Ir a la carpeta del frontend: cd somos-credito-frontend
2. Instalar dependencias: npm install
3. Verificar que la URL de la API (BASE_URL) en sucursalesApi.js apunte al backend.
4. Ejecutar en modo desarrollo: npm run dev
5. Acceder desde el navegador a la URL que muestre Vite, por ejemplo: http://localhost:5173

```js
import axios from "axios";

const API_URL = "http://localhost:3001/api/sucursales";

export const getSucursales = () => {
  return axios.get(API_URL);
};

export const createSucursal = (data) => {
  return axios.post(API_URL, data);
};

export const updateSucursal = (id, data) => {
  return axios.put(`${API_URL}/${id}`, data);
};

export const deleteSucursal = (id) => {
  return axios.delete(`${API_URL}/${id}`);
};

// cambiar solo estado (alta / baja lógica)
export const updateEstadoSucursal = (sucursal, nuevoEstado) => {
  return axios.put(`${API_URL}/${sucursal.id}`, {
    ...sucursal,
    estado: nuevoEstado,
  });
};

```

9. Conclusiones

El sistema desarrollado cumple con los requerimientos principales:

- Permite administrar la información de sucursales mediante un **CRUD completo**.
- Facilita la **búsqueda y filtrado** de información.
- Mejora la experiencia del usuario con **paginación**, **alertas interactivas** y **exportación a PDF**.
- Utiliza una **arquitectura separada** entre frontend y backend, favoreciendo la mantenibilidad.