script.js

```
document.addEventListener('DOMContentLoaded', function() {
    // Dados de exemplo (em um projeto real, isso viria de um banco de dados)
    const sampleData = {
        members: [
            {
                id: 1,
                name: "João Silva",
                age: 22,
                gender: "Masculino",
                contact: "joao@email.com | (11) 99999-9999",
                maritalStatus: "Solteiro",
                role: "Líder de Louvor",
                occupation: "Estudante",
                company: "Universidade XYZ",
                availability: "Finais de semana",
                socialMedia: "@joaosilva"
            },
            // Adicione mais membros conforme necessário
        ],
        newMembers: [
            {
                id: 101,
                name: "Maria Oliveira",
                entryDate: "2023-03-15",
        invitedBy: "João Silva",

        firstImpression: "Muito interessada em servir",

        interests: "Louvor, Eventos sociais",

        discipleshipNeed: "Médio"

    }

],

inactiveMembers: [

    {

        id: 201,

        name: "Carlos Souza",

        lastParticipation: "2023-01-10",

        reason: "Mudou de cidade",

        contactAttempts: "Sim - 2023-02-15 - Ligação sem resposta",

        responsible: "Ana Paula"

    }

],

contributions: [

    {

        id: 301,

        memberName: "João Silva",

        eventParticipation: 5,

        financialDonations: 150.00,

        timeDonations: 20,

        engagementComments: "Muito comprometido"

    }

],
projects: [
  {
    id: 401,
    name: "Retiro Espiritual",
    responsible: "Ana Paula",
    team: "João, Maria, Carlos",
    startDate: "2023-04-01",
    deadline: "2023-04-30",
    status: "Em andamento",
    results: "Espera-se 50 participantes"
  }
],
schedule: [
  {
    id: 501,
    date: "2023-03-20",
    event: "Estudo Bíblico",
    responsible: "Pastor Marcos",
    notes: "Trazer Bíblias"
  }
],
feedback: [
  {
    id: 601,
    name: "João Silva",
    comment: "Adorei o último encontro! Sugiro mais dinâmicas em grupo.",
            rating: 9,
            date: "2023-03-10",
            improvements: "Mais tempo para compartilhar experiências"
        }
    ]
};


// Inicializa os dados no localStorage se não existirem
if (!localStorage.getItem('churchYouthGroupData')) {
    localStorage.setItem('churchYouthGroupData', JSON.stringify(sampleData));
}


// Carrega os dados
let appData = JSON.parse(localStorage.getItem('churchYouthGroupData'));


// Elementos da UI
const tabButtons = document.querySelectorAll('.tab-btn');
const tabPanes = document.querySelectorAll('.tab-pane');
const searchInput = document.getElementById('search-input');
const filterSelect = document.getElementById('filter-select');
const addButton = document.getElementById('add-btn');
const modal = document.getElementById('modal');
const modalTitle = document.getElementById('modal-title');
const formFields = document.getElementById('form-fields');

const cancelButton = document.getElementById('cancel-btn');

const saveButton = document.getElementById('save-btn');

const dataForm = document.getElementById('data-form');

const exportBtn = document.getElementById('export-btn');

const importBtn = document.getElementById('import-btn');



// Variáveis de estado

let currentTab = 'members';

let currentEditId = null;

let currentEditType = null;



// Inicializa a aplicação

function initApp() {

    setupEventListeners();

    renderCurrentTab();

    updateDashboardCharts();

}



// Configura os event listeners

function setupEventListeners() {

    // Navegação por abas

    tabButtons.forEach(button => {
```

```
    button.addEventListener('click', () => {

        currentTab = button.getAttribute('data-tab');

        updateActiveTab();

        renderCurrentTab();

    });

});



// Busca

searchInput.addEventListener('input', () => {

    renderCurrentTab();

});



// Filtro

filterSelect.addEventListener('change', () => {

    renderCurrentTab();

});



// Botão adicionar

addButton.addEventListener('click', () => {

    openAddModal();

});
```

```
    // Modal
    cancelButton.addEventListener('click', closeModal);
    document.querySelector('.close-btn').addEventListener('click', closeModal);


    // Formulário
    dataForm.addEventListener('submit', (e) => {
        e.preventDefault();
        saveData();
    });


    // Exportar/Importar
    exportBtn.addEventListener('click', exportData);
    importBtn.addEventListener('click', () => {
        document.getElementById('import-file').click();
    });


    document.getElementById('import-file').addEventListener('change', importData);
}


// Atualiza a aba ativa
function updateActiveTab() {
    tabButtons.forEach(button => {
        button.classList.remove('active');
        if (button.getAttribute('data-tab') === currentTab) {
            button.classList.add('active');
        }
    });


    tabPanes.forEach(pane => {
        pane.classList.remove('active');
        if (pane.id === currentTab) {
            pane.classList.add('active');
        }
    });


    // Atualiza o texto do botão Adicionar
    updateAddButtonText();
}


// Atualiza o texto do botão Adicionar
function updateAddButtonText() {
    const texts = {
        'members': 'Membro',
        'new-members': 'Novo Membro',
        'inactive': 'Registro',
            'contributions': 'Contribuição',

            'projects': 'Projeto',

            'schedule': 'Evento',

            'feedback': 'Feedback'

        };



        addButton.innerHTML = `<i class="fas fa-plus"></i> Adicionar ${texts[currentTab] ||
'Item'}`;

    }



    // Renderiza a aba atual

    function renderCurrentTab() {

        const searchTerm = searchInput.value.toLowerCase();

        const filterValue = filterSelect.value;



        switch (currentTab) {

            case 'members':

                renderMembersTable(searchTerm, filterValue);

                break;

            case 'new-members':

                renderNewMembersTable(searchTerm, filterValue);

                break;

            case 'inactive':

                renderInactiveTable(searchTerm, filterValue);
            break;
        case 'contributions':
            renderContributionsTable(searchTerm, filterValue);
            updateContributionStats();
            break;
        case 'projects':
            renderProjectsTable(searchTerm, filterValue);
            break;
        case 'schedule':
            renderCalendar();
            break;
        case 'feedback':
            renderFeedbackTable(searchTerm, filterValue);
            updateFeedbackStats();
            break;
        case 'dashboard':
            updateDashboardCharts();
            break;
    }
}


// Renderiza a tabela de membros
function renderMembersTable(searchTerm = '', filter = 'all') {
    const tableBody = document.querySelector('#members-table tbody');
    tableBody.innerHTML = '';
```

```
let filteredData = appData.members;



// Aplica filtro
if (filter === 'active') {
    // Filtro para membros ativos (exemplo)
    filteredData = filteredData.filter(member =>
        !appData.inactiveMembers.some(inactive => inactive.name === member.name)
    );
}



// Aplica busca
if (searchTerm) {
    filteredData = filteredData.filter(member =>
        member.name.toLowerCase().includes(searchTerm) ||
        (member.role && member.role.toLowerCase().includes(searchTerm)) ||
        (member.occupation && member.occupation.toLowerCase().includes(searchTerm))
    );
}



// Preenche a tabela
filteredData.forEach(member => {
        const row = document.createElement('tr');


        // Destaca aniversariantes (exemplo)

        const isBirthday = Math.random() > 0.8; // Simulação - substitua por lógica real

        if (isBirthday) {

            row.classList.add('birthday-row');

        }




        row.innerHTML = `

            <td>${member.name}</td>

            <td>${member.age}</td>

            <td>${member.gender}</td>

            <td>${member.contact}</td>

            <td>${member.maritalStatus}</td>

            <td>${member.role || '-'}</td>

            <td>${member.occupation}</td>

            <td>${member.availability}</td>

            <td>

                <button class="action-btn edit" data-id="${member.id}"><i class="fas fa-edit"></i>
</button>

                <button class="action-btn delete" data-id="${member.id}"><i class="fas fa-trash"></i>
</button>

            </td>

        `;
        tableBody.appendChild(row);
    });



    // Adiciona event listeners para os botões de ação
    addEditDeleteListeners('members');
}



// Renderiza a tabela de novos membros
function renderNewMembersTable(searchTerm = '', filter = 'all') {
    const tableBody = document.querySelector('#new-members-table tbody');
    tableBody.innerHTML = '';



    let filteredData = appData.newMembers;



    // Aplica busca
    if (searchTerm) {
        filteredData = filteredData.filter(member =>
            member.name.toLowerCase().includes(searchTerm) ||
            (member.invitedBy && member.invitedBy.toLowerCase().includes(searchTerm))
        );
    }
    // Preenche a tabela
    filteredData.forEach(member => {
        const row = document.createElement('tr');

        // Define a cor com base na necessidade de discipulado
        let discipleshipClass = '';
        if (member.discipleshipNeed === 'Alto') discipleshipClass = 'high-need';
        else if (member.discipleshipNeed === 'Médio') discipleshipClass = 'medium-need';
        else if (member.discipleshipNeed === 'Baixo') discipleshipClass = 'low-need';


        row.innerHTML = `
            <td>${member.name}</td>
            <td>${formatDate(member.entryDate)}</td>
            <td>${member.invitedBy}</td>
            <td>${member.interests}</td>
            <td class="${discipleshipClass}">${member.discipleshipNeed}</td>
            <td>
                <button class="action-btn edit" data-id="${member.id}"><i class="fas fa-edit"></i>
</button>
                <button class="action-btn delete" data-id="${member.id}"><i class="fas fa-trash"></i>
</button>
            </td>
        `;
        tableBody.appendChild(row);
    });

    addEditDeleteListeners('new-members');
}

// Renderiza a tabela de membros inativos
function renderInactiveTable(searchTerm = '', filter = 'all') {
    const tableBody = document.querySelector('#inactive-table tbody');
    tableBody.innerHTML = '';

    let filteredData = appData.inactiveMembers;

    // Aplica busca
    if (searchTerm) {
        filteredData = filteredData.filter(member =>
            member.name.toLowerCase().includes(searchTerm) ||
            (member.responsible && member.responsible.toLowerCase().includes(searchTerm))
        );
    }
```

```
        // Preenche a tabela
        filteredData.forEach(member => {
          const row = document.createElement('tr');

          // Define a cor com base no tempo sem contato
          const lastContact = new Date(member.lastParticipation);
          const daysInactive = Math.floor((new Date() - lastContact) / (1000 * 60 * 60 * 24));
          let inactivityClass = '';

          if (daysInactive > 60) inactivityClass = 'long-inactive';
          else if (daysInactive > 30) inactivityClass = 'medium-inactive';
          else inactivityClass = 'recent-inactive';


          row.innerHTML = `
            <td>${member.name}</td>
            <td>${formatDate(member.lastParticipation)} (${daysInactive} dias)</td>
            <td>${member.reason || 'Não informado'}</td>
            <td>${member.contactAttempts || 'Nenhuma'}</td>
            <td>${member.responsible}</td>
            <td>
              <button class="action-btn edit" data-id="${member.id}"><i class="fas fa-edit"></i>
</button>
              <button class="action-btn delete" data-id="${member.id}"><i class="fas fa-trash"></i>
</button>
```

```
            </td>
        `;


        row.classList.add(inactivityClass);
        tableBody.appendChild(row);
    });


    addEditDeleteListeners('inactive');
}


// Renderiza a tabela de contribuições
function renderContributionsTable(searchTerm = '', filter = 'all') {
    const tableBody = document.querySelector('#contributions-table tbody');
    tableBody.innerHTML = '';


    let filteredData = appData.contributions;


    // Aplica busca
    if (searchTerm) {
        filteredData = filteredData.filter(contribution =>
            contribution.memberName.toLowerCase().includes(searchTerm)
        );
    }


    // Preenche a tabela
    filteredData.forEach(contribution => {
        const row = document.createElement('tr');

        // Calcula engajamento (exemplo simplificado)
        const engagementScore = Math.min(10, Math.floor(
            (contribution.eventParticipation * 0.4) +
            (contribution.timeDonations * 0.3) +
            (contribution.financialDonations / 50 * 0.3)
        ));


        let engagementClass = '';
        if (engagementScore >= 8) engagementClass = 'high-engagement';
        else if (engagementScore >= 5) engagementClass = 'medium-engagement';
        else engagementClass = 'low-engagement';


        row.innerHTML = `
            <td>${contribution.memberName}</td>
            <td>${contribution.eventParticipation}</td>
            <td>R$ ${contribution.financialDonations.toFixed(2)}</td>
```

```
                <td>${contribution.timeDonations}h</td>

                <td class="${engagementClass}">

                    ${'★'.repeat(engagementScore)}${'☆'.repeat(10 - engagementScore)}

                </td>

                <td>

                    <button class="action-btn edit" data-id="${contribution.id}"><i class="fas fa-edit"></i>
</button>

                    <button class="action-btn delete" data-id="${contribution.id}"><i class="fas fa-trash">
</i></button>

                </td>

            `;



            tableBody.appendChild(row);

        });



        addEditDeleteListeners('contributions');

    }



    // Atualiza as estatísticas de contribuição

    function updateContributionStats() {

        const totalHours = appData.contributions.reduce((sum, item) => sum + item.timeDonations,
0);

        const totalDonations = appData.contributions.reduce((sum, item) => sum +
item.financialDonations, 0);
```

```
        const totalEvents = appData.contributions.reduce((sum, item) => sum +
item.eventParticipation, 0);

        }



        document.getElementById('total-hours').textContent = totalHours;

        document.getElementById('total-donations').textContent = `R$
${totalDonations.toFixed(2)}`;

        document.getElementById('total-events').textContent = totalEvents;

    }



    // Renderiza a tabela de projetos

    function renderProjectsTable(searchTerm = '', filter = 'all') {

        const tableBody = document.querySelector('#projects-table tbody');

        tableBody.innerHTML = '';



        let filteredData = appData.projects;



        // Aplica filtro

        if (filter !== 'all') {

            filteredData = filteredData.filter(project =>

                project.status.toLowerCase().includes(filter.toLowerCase())

            );

        }
```

```
// Aplica busca
if (searchTerm) {
    filteredData = filteredData.filter(project =>
        project.name.toLowerCase().includes(searchTerm) ||
        (project.responsible && project.responsible.toLowerCase().includes(searchTerm))
    );
}


// Preenche a tabela
filteredData.forEach(project => {
    const row = document.createElement('tr');

    // Define a classe com base no status
    let statusClass = '';
    if (project.status === 'Concluído') statusClass = 'completed';
    else if (project.status === 'Em andamento') statusClass = 'in-progress';
    else if (project.status === 'Pendente') statusClass = 'pending';


    // Calcula progresso (exemplo simplificado)
    const startDate = new Date(project.startDate);
    const endDate = new Date(project.deadline);
    const today = new Date();
```

```
        let progress = 0;
        if (today >= endDate) progress = 100;
        else if (today > startDate) {
            const totalDays = (endDate - startDate) / (1000 * 60 * 60 * 24);
            const daysPassed = (today - startDate) / (1000 * 60 * 60 * 24);
            progress = Math.min(99, Math.floor((daysPassed / totalDays) * 100));
        }


        row.innerHTML = `
            <td>${project.name}</td>
            <td>${project.responsible}</td>
            <td>${project.team}</td>
            <td>${formatDate(project.startDate)}</td>
            <td>${formatDate(project.deadline)}</td>
            <td class="${statusClass}">
                <div class="progress-container">
                    <div class="progress-bar" style="width: ${progress}%"></div>
                    <span>${project.status} (${progress}%)</span>
                </div>
            </td>
            <td>
                <button class="action-btn edit" data-id="${project.id}"><i class="fas fa-edit"></i>
</button>
                <button class="action-btn delete" data-id="${project.id}"><i class="fas fa-trash"></i>
</button>
```

```
            </td>
        `;

        tableBody.appendChild(row);
    });

    addEditDeleteListeners('projects');
}

// Renderiza o calendário
function renderCalendar() {
    const calendarGrid = document.getElementById('calendar');
    calendarGrid.innerHTML = '';

    const currentDate = new Date();
    const currentMonth = currentDate.getMonth();
    const currentYear = currentDate.getFullYear();

    // Atualiza o cabeçalho do mês
    document.getElementById('current-month').textContent =
        `${getMonthName(currentMonth)} ${currentYear}`;
```

```
// Primeiro dia do mês

const firstDay = new Date(currentYear, currentMonth, 1);

// Último dia do mês

const lastDay = new Date(currentYear, currentMonth + 1, 0);


// Dias da semana

const weekdays = ['Dom', 'Seg', 'Ter', 'Qua', 'Qui', 'Sex', 'Sáb'];


// Adiciona cabeçalhos dos dias da semana

weekdays.forEach(day => {

    const dayHeader = document.createElement('div');

    dayHeader.className =
```