<!DOCTYPE html>

<html lang="pt-BR">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0, user-scalable=no">

<title>ACE FLOW - Agentes de Combate às Endemias</title>

<!-- Dependências -->

<script src="https://cdn.tailwindcss.com"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/jspdf/2.5.1/jspdf.umd.min.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/jspdf-autotable/3.8.2/jspdf.plugin.autotable.min.js"></script>

<script src="https://cdn.jsdelivr.net/npm/chart.js@4.4.2/dist/chart.umd.min.js"></script>

<link rel="preconnect" href="https://fonts.googleapis.com">

<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>

<link href="https://fonts.googleapis.com/css2?family=Inter:wght@400;500;600;700;800&display=swap" rel="stylesheet">

<!-- PWA (Progressive Web App) Manifest & Service Worker -->

<meta name="theme-color" content="#C3B091">

<script>

document.addEventListener('DOMContentLoaded', () => {

const manifest = {

"name": "ACE FLOW",

"short\_name": "ACE FLOW",

"start\_url": ".",

"display": "standalone",

"background\_color": "#F5F0E6",

"theme\_color": "#C3B091",

"description": "App para registro de trabalho de Agentes de Combate às Endemias.",

"icons": [{ "src": "https://i.ibb.co/6wmz9wN/ace-flow-logo-192.png", "sizes": "192x192", "type": "image/png" }]

};

const manifestString = JSON.stringify(manifest);

const blob = new Blob([manifestString], {type: 'application/json'});

const manifestURL = URL.createObjectURL(blob);

const linkTag = document.createElement('link');

linkTag.rel = 'manifest';

linkTag.href = manifestURL;

document.head.appendChild(linkTag);

if ('serviceWorker' in navigator) {

const swCode = `

const CACHE\_NAME = 'ace-flow-cache-v11';

self.addEventListener('install', event => {

event.waitUntil(

caches.open(CACHE\_NAME).then(cache => {

return fetch(new Request(self.location.href)).then(response => {

return cache.put(self.location.href, response);

});

})

);

});

self.addEventListener('fetch', event => {

if (event.request.mode === 'navigate') {

event.respondWith(

caches.match(event.request).then(response => {

return response || fetch(event.request);

})

);

}

});

`;

const swBlob = new Blob([swCode], { type: 'application/javascript' });

const swUrl = URL.createObjectURL(swBlob);

window.addEventListener('load', () => {

navigator.serviceWorker.register(swUrl)

.then(reg => console.log('ServiceWorker registrado.', reg.scope))

.catch(err => console.log('ServiceWorker falhou:', err));

});

}

});

</script>

<!-- Estilos -->

<style>

:root {

--bg-main: #F5F0E6;

--bg-card: #FFFFFF;

--text-dark: #000000;

--text-muted: #575757;

--border-color: #DED6C7;

--alert-red: #C1443D;

--confirm-green: #16A34A;

--khaki-base: #C3B091;

--khaki-dark: #a9997c;

}

body { font-family: 'Inter', sans-serif; background-color: var(--bg-main); color: var(--text-dark); }

.card { background-color: var(--bg-card); border-radius: 0.75rem; box-shadow: 0 4px 12px rgba(0,0,0,0.08); border: 1px solid var(--border-color); }

.section-title { font-weight: 700; font-size: 1.25rem; color: var(--text-dark); }

/\* Formulários \*/

.form-input, .form-select {

width: 100%; padding: 0.75rem; border-radius: 0.5rem; border: 2px solid var(--border-color);

background-color: var(--bg-card); transition: all 0.2s ease;

}

.form-input:focus, .form-select:focus {

outline: none; border-color: var(--khaki-base); box-shadow: 0 0 0 3px rgba(195, 176, 145, 0.4);

}

.form-input.is-filled, .form-select.is-filled {

border-color: var(--confirm-green);

}

/\* Botões \*/

.btn {

padding: 0.8rem 1.25rem; border-radius: 0.5rem; font-weight: 700; transition: all 0.2s;

display: flex; align-items: center; justify-content: center; gap: 0.5rem; cursor: pointer; border: none;

box-shadow: 0 2px 4px rgba(0,0,0,0.1);

}

.btn:disabled { background-color: #9ca3af; cursor: not-allowed; opacity: 0.7; }

.btn-primary { background-color: var(--text-dark); color: white; }

.btn-primary:hover:not(:disabled) { background-color: #333; }

.btn-secondary { background-color: var(--khaki-base); color: white; }

.btn-secondary:hover:not(:disabled) { background-color: var(--khaki-dark); }

.btn-danger { background-color: var(--alert-red); color: white; }

/\* Acordeão e Conteúdo Oculto \*/

.hidden-content { display: none; overflow: hidden; transition: max-height 0.5s ease-in-out; max-height: 0; }

.hidden-content.show { display: block; max-height: 1500px; }

/\* Botões de Ação Especiais (Trat, Foco) \*/

.toggle-check {

width: 100%; height: 46px; font-weight: 700; font-size: 1.1rem; cursor: pointer;

border: 2px solid var(--border-color); border-radius: 0.5rem; background-color: var(--bg-card);

transition: all 0.2s; display: flex; align-items: center; justify-content: center;

}

.toggle-check.tratado.is-checked { background-color: #dcfce7; color: var(--confirm-green); border-color: var(--confirm-green); }

.toggle-check.foco.is-checked { background-color: #fee2e2; color: var(--alert-red); border-color: var(--alert-red); }

/\* Barra de Informação Fixa \*/

#fixed-info-bar {

position: sticky; top: 0; z-index: 40; transition: all 0.3s ease;

box-shadow: 0 4px 6px -1px rgba(0, 0, 0, 0.1), 0 2px 4px -2px rgba(0, 0, 0, 0.1);

}

/\* Modal \*/

.modal {

background-color: rgba(0, 0, 0, 0.7); display: flex;

align-items: center; justify-content: center;

}

/\* Opções de Larvicida em Negrito \*/

option.important-value {

font-weight: 700;

background-color: #eeeeee;

}

</style>

</head>

<body class="antialiased">

<!-- BARRA DE INFORMAÇÃO FIXA -->

<div id="fixed-info-bar" class="bg-white/80 backdrop-blur-sm p-2 text-center text-sm font-semibold hidden">

<span id="fixed-agent-name"></span> - <span id="fixed-property-info"></span>

</div>

<!-- MODAL DE LOGIN -->

<div id="loginModal" class="fixed inset-0 z-50 modal">

<div class="bg-white rounded-lg shadow-xl p-8 w-11/12 max-w-sm card">

<h2 class="text-2xl font-bold text-center mb-1">Bem-vindo ao</h2>

<h1 class="text-4xl font-extrabold text-center mb-6">ACE FLOW</h1>

<form id="loginForm">

<div class="mb-4">

<label for="username" class="block text-sm font-medium text-text-muted mb-1">Usuário</label>

<input type="text" id="username" class="form-input" required>

</div>

<div class="mb-6">

<label for="password" class="block text-sm font-medium text-text-muted mb-1">Senha</label>

<input type="password" id="password" class="form-input" required>

</div>

<button type="submit" id="loginBtn" class="btn btn-primary w-full text-lg">Entrar</button>

<p id="loginError" class="text-red-600 text-center mt-4 h-4"></p>

</form>

</div>

</div>

<main class="container mx-auto p-3 md:p-6 max-w-4xl" style="display: none;">

<!-- CABEÇALHO -->

<header class="text-center mb-8 py-4">

<h1 class="text-5xl font-extrabold"><strong>ACE FLOW</strong></h1>

<p class="text-lg font-bold">Agentes de Combate às Endemias</p>

<p class="text-md"><em>Registre aqui a excelência do seu trabalho.</em></p>

</header>

<!-- MÓDULO: MEU SETOR & META DIÁRIA -->

<section class="card p-4 md:p-6 mb-6">

<details id="goalModule" class="group">

<summary class="section-title cursor-pointer flex justify-between items-center">

Meu Setor & Meta Diária

<span class="text-2xl font-bold text-khaki-base transition-transform duration-300 group-open:rotate-45">+</span>

</summary>

<div class="mt-4 border-t pt-4">

<div class="grid grid-cols-1 md:grid-cols-3 gap-4 mb-4">

<div>

<label for="totalImoveis" class="text-sm font-medium text-text-muted">Total de Imóveis</label>

<input type="number" id="totalImoveis" class="form-input mt-1" placeholder="Ex: 800">

</div>

<div>

<label for="cicloInicio" class="text-sm font-medium text-text-muted">Início do Ciclo</label>

<input type="date" id="cicloInicio" class="form-input mt-1">

</div>

<div>

<label for="cicloFim" class="text-sm font-medium text-text-muted">Fim do Ciclo</label>

<input type="date" id="cicloFim" class="form-input mt-1">

</div>

</div>

<button id="calculateGoalBtn" class="btn btn-secondary w-full">Calcular Meta</button>

<div id="goalResult" class="mt-4 text-center hidden">

<p class="text-lg">Meta diária: <strong id="dailyGoal" class="text-xl">0</strong> imóveis</p>

<div class="w-full bg-gray-200 rounded-full h-4 mt-2 border border-border-color">

<div id="dailyProgress" class="bg-confirm-green h-full rounded-full text-xs font-medium text-blue-100 text-center p-0.5 leading-none" style="width: 0%"></div>

</div>

<p id="progressText" class="text-sm text-text-muted mt-1">0/0</p>

</div>

</div>

</details>

</section>

<!-- MÓDULO: INFORMAÇÕES DO DIA -->

<section class="card p-4 md:p-6 mb-6">

<h2 class="section-title border-b border-border-color pb-3 mb-4">Informações do Dia</h2>

<div class="grid grid-cols-1 md:grid-cols-2 gap-4">

<div>

<label for="agente" class="text-sm font-medium text-text-muted">Agente</label>

<input type="text" id="agente" list="agent-suggestions" class="form-input mt-1" required>

</div>

<div>

<label for="bairro" class="text-sm font-medium text-text-muted">Bairro</label>

<input type="text" id="bairro" list="bairro-suggestions" class="form-input mt-1" required>

</div>

<div>

<label for="ciclo" class="text-sm font-medium text-text-muted">Ciclo</label>

<select id="ciclo" class="form-select mt-1" required></select>

</div>

<div>

<label for="data" class="text-sm font-medium text-text-muted">Data</label>

<input type="date" id="data" class="form-input mt-1">

</div>

</div>

</section>

<!-- MÓDULO: REGISTROS DE VISITA -->

<section class="space-y-4">

<h2 class="section-title text-center mt-8 mb-4">Registros de Visita</h2>

<div id="visitasContainer" class="space-y-3"></div>

<button id="addVisitaBtn" class="btn btn-primary w-full text-lg">

<svg xmlns="http://www.w3.org/2000/svg" width="20" height="20" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="3" stroke-linecap="round" stroke-linejoin="round"><line x1="12" y1="5" x2="12" y2="19"></line><line x1="5" y1="12" x2="19" y2="12"></line></svg>

Adicionar Imóvel

</button>

</section>

<!-- MÓDULO: RESUMO GERAL -->

<section class="card p-4 md:p-6 my-8">

<h2 class="section-title border-b border-border-color pb-3 mb-4">Resumo Geral</h2>

<div id="summarySection" class="grid grid-cols-2 md:grid-cols-4 gap-x-4 gap-y-5"></div>

</section>

<!-- BOTÕES DE AÇÃO -->

<footer class="grid grid-cols-2 md:grid-cols-4 gap-4 my-6">

<button id="reportBtn" class="btn btn-secondary">Relatório</button>

<button id="clearDayBtn" class="btn btn-danger">Limpar Dia</button>

<button id="savePdfBtn" class="btn btn-primary">Gerar PDF</button>

<button id="saveRemoteBtn" class="btn btn-primary">Enviar p/ Planilha</button>

</footer>

</main>

<!-- DATALISTS E MODAIS -->

<datalist id="agent-suggestions"></datalist>

<datalist id="bairro-suggestions"></datalist>

<datalist id="street-suggestions"></datalist>

<div id="confirmationModal" class="fixed inset-0 z-50 modal hidden">

<div class="card p-6 w-11/12 max-w-sm text-center">

<p id="modalMessage" class="text-lg mb-6">Tem certeza?</p>

<div class="flex justify-center gap-4">

<button id="modalCancelBtn" class="btn btn-secondary">Cancelar</button>

<button id="modalConfirmBtn" class="btn btn-primary">Confirmar</button>

</div>

</div>

</div>

<div id="reportModal" class="fixed inset-0 z-50 modal hidden">

<div class="card p-4 md:p-6 w-11/12 max-w-4xl max-h-[90vh] flex flex-col">

<header class="flex justify-between items-center border-b pb-3 mb-4">

<h2 class="section-title">Relatório de Atividades</h2>

<button id="closeReportModalBtn" class="text-3xl font-bold">&times;</button>

</header>

<div class="flex-grow overflow-y-auto">

<p class="text-center my-8 text-text-muted">Funcionalidade de Relatórios e Gráficos em desenvolvimento.</p>

<!-- Filtros (a implementar) -->

<!-- Gráfico (a implementar) -->

<canvas id="reportChart" class="hidden"></canvas>

<!-- Tabela (a implementar) -->

<div id="reportTableContainer"></div>

</div>

</div>

</div>

<script>

document.addEventListener('DOMContentLoaded', () => {

// --- VARIÁVEIS GLOBAIS ---

const SCRIPT\_URL = 'https://script.google.com/macros/s/AKfycbx\_M9m5XNbdgaMGXF69esRYPT-xBu5yVz4b6TqUrvy85Ipx\_hVh4fQUqISwtFhoMkEW/exec';

const { jsPDF } = window.jspdf;

const Chart = window.Chart;

const INITIAL\_ROWS = 15;

const CICLOS = ['1°', '2°', '3°', '4°', '5°', '6°'];

const BRAZIL\_HOLIDAYS\_2025 = [ // Feriados Nacionais Fixos e Móveis de 2025

'2025-01-01', '2025-03-03', '2025-03-04', '2025-04-18', '2025-04-21',

'2025-05-01', '2025-06-19', '2025-09-07', '2025-10-12', '2025-11-02',

'2025-11-15', '2025-11-20', '2025-12-25',

];

let streetList = [], agentList = [], bairroList = [];

let visitaCounter = 0;

let confirmCallback = null;

let dailyGoal = 0;

let myChart;

// --- SELETORES DO DOM ---

const mainContent = document.querySelector('main');

const loginModal = document.getElementById('loginModal');

const loginForm = document.getElementById('loginForm');

const loginBtn = document.getElementById('loginBtn');

const loginError = document.getElementById('loginError');

const agenteInput = document.getElementById('agente');

const bairroInput = document.getElementById('bairro');

const cicloSelect = document.getElementById('ciclo');

const dataInput = document.getElementById('data');

const visitasContainer = document.getElementById('visitasContainer');

const summarySection = document.getElementById('summarySection');

const agentSuggestions = document.getElementById('agent-suggestions');

const bairroSuggestions = document.getElementById('bairro-suggestions');

const streetSuggestions = document.getElementById('street-suggestions');

const confirmationModal = document.getElementById('confirmationModal');

const modalMessage = document.getElementById('modalMessage');

const modalConfirmBtn = document.getElementById('modalConfirmBtn');

const modalCancelBtn = document.getElementById('modalCancelBtn');

const reportModal = document.getElementById('reportModal');

const reportBtn = document.getElementById('reportBtn');

const closeReportModalBtn = document.getElementById('closeReportModalBtn');

const fixedInfoBar = document.getElementById('fixed-info-bar');

const fixedAgentName = document.getElementById('fixed-agent-name');

const fixedPropertyInfo = document.getElementById('fixed-property-info');

const totalImoveisInput = document.getElementById('totalImoveis');

const cicloInicioInput = document.getElementById('cicloInicio');

const cicloFimInput = document.getElementById('cicloFim');

const calculateGoalBtn = document.getElementById('calculateGoalBtn');

const goalResultDiv = document.getElementById('goalResult');

const dailyGoalEl = document.getElementById('dailyGoal');

const dailyProgressEl = document.getElementById('dailyProgress');

const progressTextEl = document.getElementById('progressText');

// --- FUNÇÕES DE UTILIDADE ---

const toTitleCase = (str) => {

if (!str) return '';

const lower = ['de', 'da', 'do', 'dos', 'das', 'e'];

return str.replace(/\w\S\*/g, (txt, offset) => {

const word = txt.toLowerCase();

if (offset > 0 && lower.includes(word)) {

return word;

}

return txt.charAt(0).toUpperCase() + txt.substr(1).toLowerCase();

});

};

const corrigirOrtografia = (str) => {

if (!str) return '';

const correcoes = {

'sao': 'São', 'sebastiao': 'Sebastião', 'jose': 'José',

'joao': 'João', 'conceicao': 'Conceição', 'vitoria': 'Vitória',

'praca': 'Praça', 'acude': 'Açude', 'antonio': 'Antônio',

'maria': 'Maria', 'francisco': 'Francisco', 'francisca': 'Francisca',

'luiz': 'Luiz', 'gonzaga': 'Gonzaga', 'pereira': 'Pereira',

'silva': 'Silva', 'souza': 'Souza', 'santos': 'Santos',

'oliveira': 'Oliveira', 'tv': 'Tv', 'av': 'Av', 'r': 'Rua',

'trav': 'Travessa', 'sen': 'Senador'

};

let correctedStr = str;

Object.keys(correcoes).forEach(key => {

const regex = new RegExp(`\\b${key}\\b`, 'gi');

correctedStr = correctedStr.replace(regex, correcoes[key]);

});

return correctedStr;

};

const formatAndCorrectText = (str) => {

return toTitleCase(corrigirOrtografia(str));

}

// --- TEMPLATES HTML ---

const createVisitaCardHTML = (index) => `

<div class="card visita-card" data-index="${index}">

<header class="flex items-center justify-between p-3 cursor-pointer card-header group" data-toggle="card-content-${index}">

<div class="flex items-center min-w-0 flex-1">

<span class="flex-shrink-0 flex items-center justify-center h-8 w-8 rounded-full bg-bg-card border-2 border-khaki-base text-khaki-base font-bold mr-3">${index + 1}</span>

<div class="min-w-0 flex-grow">

<span class="logradouro-preview font-semibold truncate block">Novo Imóvel</span>

<div class="status-indicator text-sm font-semibold text-alert-red">Falta Preencher</div>

</div>

</div>

<span class="toggle-icon text-3xl font-bold text-khaki-base transition-transform duration-300 group-data-[expanded=true]:rotate-45">+</span>

</header>

<div id="card-content-${index}" class="hidden-content">

<div class="p-4 border-t border-border-color space-y-4">

<div>

<label class="text-sm font-bold text-text-muted">Logradouro/Rua</label>

<input type="text" list="street-suggestions" class="form-input logradouro mt-1" placeholder="Ex: Rua João da Silva">

</div>

<div class="grid grid-cols-3 gap-x-3">

<div><label class="text-sm font-bold text-text-muted">Quarteirão</label><input type="number" min="1" class="form-input quarteirao mt-1 text-center"></div>

<div><label class="text-sm font-bold text-text-muted">Lado</label><select class="form-select lado mt-1" required><option value="" disabled selected></option><option value="1">1</option><option value="2">2</option><option value="3">3</option><option value="4">4</option></select></div>

<div><label class="text-sm font-bold text-text-muted">Nº</label><input type="text" class="form-input numero mt-1 text-center is-filled" value="S/N" onfocus="if(this.value==='S/N')this.value=''" onblur="if(this.value.trim()===''){this.value='S/N'; this.classList.add('is-filled');} else {this.classList.add('is-filled');}"></div>

</div>

<div class="grid grid-cols-2 gap-x-3">

<div><label class="text-sm font-bold text-text-muted">Imóvel</label><select class="form-select imovel mt-1" required><option value="" disabled selected></option><option value="R">Residência</option><option value="C">Comércio</option><option value="TB">Terreno Baldio</option><option value="O">Outro</option></select></div>

<div><label class="text-sm font-bold text-text-muted">Visita</label><select class="form-select visita mt-1" required><option value="" disabled selected></option><option value="T">Trabalhado</option><option value="P">Recuperado</option><option value="F">Fechado</option><option value="R">Recusado</option></select></div>

</div>

<div class="border-t border-border-color pt-4 mt-4">

<p class="text-center text-sm font-bold text-text-muted mb-2">Resultados</p>

<div class="grid grid-cols-5 gap-x-2">

<div><label class="text-xs font-medium text-center block">Elim.</label><input type="number" min="0" class="form-input elim mt-1 text-center" placeholder="0"></div>

<div><label class="text-xs font-medium text-center block">Trat.</label><button type="button" class="toggle-check tratado mt-1"></button></div>

<div><label class="text-xs font-medium text-center block">Larvicida</label><select class="form-select gram mt-1 text-center"></select></div>

<div><label class="text-xs font-medium text-center block">Dep.</label><input type="number" min="0" class="form-input dep mt-1 text-center" placeholder="0"></div>

<div><label class="text-xs font-medium text-center block">Foco</label><button type="button" class="toggle-check foco mt-1"></button></div>

</div>

</div>

</div>

</div>

</div>

`;

// --- FUNÇÕES PRINCIPAIS ---

const showModal = (message, cb) => {

modalMessage.textContent = message;

confirmCallback = cb;

confirmationModal.classList.remove('hidden');

};

const hideModal = () => {

confirmationModal.classList.add('hidden');

confirmCallback = null;

};

const updateToggleHTML = (button) => {

const isChecked = button.classList.contains('is-checked');

if (button.matches('.foco')) button.innerHTML = isChecked ? 'X' : '';

else if (button.matches('.tratado')) button.innerHTML = isChecked ? '✅' : '';

};

const populateSelects = () => {

cicloSelect.innerHTML = `<option value="" disabled selected></option>` + CICLOS.map(c => `<option value="${c}">${c}</option>`).join('');

};

const populateGramasSelect = (selectElement) => {

const importantValues = ['0.8', '1.6', '3.2', '6.4', '7.2', '9.6', '12.8'];

let gramasHTML = '<option value="0.0">0.0</option>';

for (let i = 0.4; i <= 60.0; i = parseFloat((i + 0.4).toFixed(1))) {

const val = i.toFixed(1);

const isImportant = importantValues.includes(val);

gramasHTML += `<option value="${val}" ${isImportant ? 'class="important-value"' : ''}>${val}</option>`;

}

selectElement.innerHTML = gramasHTML;

};

const addVisitaCard = () => {

const cardHTML = createVisitaCardHTML(visitaCounter);

visitasContainer.insertAdjacentHTML('beforeend', cardHTML);

const newCard = visitasContainer.lastElementChild;

populateGramasSelect(newCard.querySelector('.gram'));

if (visitaCounter === 0) newCard.querySelector('.logradouro').focus();

if(visitaCounter > 0) {

const prevCard = visitasContainer.children[visitaCounter - 1];

const prevLogradouro = prevCard.querySelector('.logradouro').value;

const prevQuarteirao = prevCard.querySelector('.quarteirao').value;

const prevLado = prevCard.querySelector('.lado').value;

if(prevLogradouro) {

newCard.querySelector('.logradouro').value = prevLogradouro;

}

if(prevQuarteirao) {

newCard.querySelector('.quarteirao').value = prevQuarteirao;

}

if(prevLado) {

newCard.querySelector('.lado').value = prevLado;

}

updateCardStatus(newCard);

}

visitaCounter++;

};

const updateCardStatus = (cardElement) => {

const logradouro = cardElement.querySelector('.logradouro').value.trim();

const numero = cardElement.querySelector('.numero').value.trim();

const preview = cardElement.querySelector('.logradouro-preview');

preview.textContent = logradouro ? `${logradouro}, ${numero}` : 'Novo Imóvel';

const inputs = ['.logradouro', '.quarteirao', '.numero', '.lado', '.imovel', '.visita'];

const isFullyFilled = inputs.every(sel => cardElement.querySelector(sel).value.trim() !== '');

const statusIndicator = cardElement.querySelector('.status-indicator');

statusIndicator.classList.remove('text-alert-red', 'text-amber-500', 'text-confirm-green');

if (isFullyFilled) {

statusIndicator.textContent = '✅ Preenchido';

statusIndicator.classList.add('text-confirm-green');

} else {

statusIndicator.textContent = 'Falta Preencher';

statusIndicator.classList.add('text-alert-red');

}

};

const updateSummary = () => {

const allCards = document.querySelectorAll('.visita-card');

const filledCards = [...allCards].filter(card => ['.logradouro', '.quarteirao', '.numero', '.lado', '.imovel', '.visita'].every(sel => card.querySelector(sel).value.trim() !== ''));

let summary = { recuperado: 0, fechado: 0, recusado: 0, eliminados: 0, depositos: 0, tratamentoFocal: 0, gramas: 0, focos: 0 };

let trabalhadosSummary = { total: 0, residencia: 0, comercio: 0, terreno: 0, outro: 0 };

const quarteiroesSet = new Set();

filledCards.forEach(card => {

const imovel = card.querySelector('.imovel').value;

const visita = card.querySelector('.visita').value;

const quarteiraoValue = card.querySelector('.quarteirao').value.trim();

if (quarteiraoValue && !isNaN(parseInt(quarteiraoValue))) {

quarteiroesSet.add(parseInt(quarteiraoValue));

}

if (visita === 'T') {

trabalhadosSummary.total++;

if (imovel === 'R') trabalhadosSummary.residencia++;

else if (imovel === 'C') trabalhadosSummary.comercio++;

else if (imovel === 'TB') trabalhadosSummary.terreno++;

else if (imovel === 'O') trabalhadosSummary.outro++;

}

else if (visita === 'P') { summary.recuperado++; }

else if (visita === 'F') { summary.fechado++; }

else if (visita === 'R') { summary.recusado++; }

summary.eliminados += parseInt(card.querySelector('.elim').value) || 0;

summary.depositos += parseInt(card.querySelector('.dep').value) || 0;

if (card.querySelector('.tratado').classList.contains('is-checked')) summary.tratamentoFocal++;

summary.gramas += parseFloat(card.querySelector('.gram').value) || 0;

if (card.querySelector('.foco').classList.contains('is-checked')) summary.focos++;

});

if(dailyGoal > 0){

const progress = Math.min((trabalhadosSummary.total / dailyGoal) \* 100, 100);

dailyProgressEl.style.width = `${progress}%`;

progressTextEl.textContent = `${trabalhadosSummary.total} / ${dailyGoal}`;

}

const quarteiroesTrabalhados = quarteiroesSet.size;

let quarteiroesConcluidos = 0;

if (quarteiroesSet.size > 1) {

const maxQuarteirao = Math.max(...quarteiroesSet);

quarteiroesConcluidos = [...quarteiroesSet].filter(q => q < maxQuarteirao).length;

}

summarySection.innerHTML = `

<div class="col-span-full text-center border-b border-border-color pb-3 mb-4">

<h3 class="font-semibold text-text-muted">Detalhes dos Imóveis Trabalhados</h3>

</div>

<div class="p-3 text-center col-span-1 summary-item"><p class="font-bold text-2xl">${trabalhadosSummary.residencia}</p><p class="text-sm text-text-muted">Residências</p></div>

<div class="p-3 text-center col-span-1 summary-item"><p class="font-bold text-2xl">${trabalhadosSummary.comercio}</p><p class="text-sm text-text-muted">Comércios</p></div>

<div class="p-3 text-center col-span-1 summary-item"><p class="font-bold text-2xl">${trabalhadosSummary.terreno}</p><p class="text-sm text-text-muted">Ter. Baldios</p></div>

<div class="p-3 text-center col-span-1 summary-item"><p class="font-bold text-2xl">${trabalhadosSummary.outro}</p><p class="text-sm text-text-muted">Outros</p></div>

<div class="p-3 text-center col-span-full font-bold text-lg bg-gray-100 rounded-lg summary-item"><p>${trabalhadosSummary.total}</p><p class="text-sm font-normal text-text-muted">Total Trabalhados</p></div>

<div class="col-span-full text-center border-t border-border-color pt-4 mt-4 pb-3 mb-4">

<h3 class="font-semibold text-text-muted">Resumo Geral da Produção</h3>

</div>

<div class="p-3 text-center col-span-1 summary-item"><p class="font-bold text-2xl">${summary.tratamentoFocal}</p><p class="text-sm text-text-muted">Trat. Focal</p></div>

<div class="p-3 text-center col-span-1 summary-item"><p class="font-bold text-2xl">${summary.gramas.toFixed(1)}g</p><p class="text-sm text-text-muted">Larvicida</p></div>

<div class="p-3 text-center col-span-1 summary-item"><p class="font-bold text-2xl">${summary.depositos}</p><p class="text-sm text-text-muted">Dep. Tratados</p></div>

<div class="p-3 text-center col-span-1 summary-item"><p class="font-bold text-2xl text-alert-red">${summary.focos}</p><p class="text-sm text-text-muted">Total de Focos</p></div>

<div class="p-3 text-center col-span-1 summary-item"><p class="font-bold text-2xl">${summary.recusado}</p><p class="text-sm text-text-muted">Recusados</p></div>

<div class="p-3 text-center col-span-1 summary-item"><p class="font-bold text-2xl">${summary.fechado}</p><p class="text-sm text-text-muted">Fechados</p></div>

<div class="p-3 text-center col-span-1 summary-item"><p class="font-bold text-2xl">${summary.recuperado}</p><p class="text-sm text-text-muted">Recuperados</p></div>

<div class="p-3 text-center col-span-1 summary-item"><p class="font-bold text-2xl">${quarteiroesTrabalhados}</p><p class="text-sm text-text-muted">Quart. Trabalhados</p></div>

<div class="p-3 text-center col-span-full summary-item"><p class="font-bold text-2xl">${quarteiroesConcluidos}</p><p class="text-sm text-text-muted">Quart. Concluídos</p></div>

`;

};

const calculateWorkingDays = (startDate, endDate) => {

let count = 0;

const curDate = new Date(startDate.getTime());

while (curDate <= endDate) {

const dayOfWeek = curDate.getUTCDay();

const dateString = curDate.toISOString().slice(0, 10);

if (dayOfWeek !== 0 && dayOfWeek !== 6 && !BRAZIL\_HOLIDAYS\_2025.includes(dateString)) count++;

curDate.setDate(curDate.getDate() + 1);

}

return count;

};

const handleGoalCalculation = () => {

const totalImoveis = parseInt(totalImoveisInput.value);

const inicio = cicloInicioInput.value;

const fim = cicloFimInput.value;

if (totalImoveis > 0 && inicio && fim) {

const workingDays = calculateWorkingDays(new Date(inicio + 'T00:00:00'), new Date(fim + 'T00:00:00'));

if(workingDays > 0) {

dailyGoal = Math.ceil(totalImoveis / workingDays);

dailyGoalEl.textContent = dailyGoal;

goalResultDiv.classList.remove('hidden');

updateSummary();

localStorage.setItem('aceFlowGoalData', JSON.stringify({ totalImoveis, inicio, fim, dailyGoal }));

} else { alert("O período selecionado não contém dias úteis."); }

} else { alert("Por favor, preencha todos os campos do setor."); }

};

const loadGoalData = () => {

const goalData = JSON.parse(localStorage.getItem('aceFlowGoalData'));

if (goalData) {

totalImoveisInput.value = goalData.totalImoveis;

cicloInicioInput.value = goalData.inicio;

cicloFimInput.value = goalData.fim;

dailyGoal = goalData.dailyGoal;

dailyGoalEl.textContent = dailyGoal;

goalResultDiv.classList.remove('hidden');

document.getElementById('goalModule').open = true;

}

};

const saveData = () => {

const data = {

header: { agente: agenteInput.value, bairro: bairroInput.value, ciclo: cicloSelect.value, data: dataInput.value },

visits: []

};

document.querySelectorAll('.visita-card').forEach(card => {

data.visits.push({

logradouro: card.querySelector('.logradouro').value, quarteirao: card.querySelector('.quarteirao').value,

lado: card.querySelector('.lado').value, numero: card.querySelector('.numero').value,

imovel: card.querySelector('.imovel').value, visita: card.querySelector('.visita').value,

elim: card.querySelector('.elim').value, trat: card.querySelector('.tratado').classList.contains('is-checked'),

gram: card.querySelector('.gram').value, dep: card.querySelector('.dep').value,

foco: card.querySelector('.foco').classList.contains('is-checked'),

isExpanded: card.querySelector('.card-header').getAttribute('data-expanded') === 'true'

});

});

localStorage.setItem('aceFlowDailyData', JSON.stringify(data));

};

const loadData = () => {

const data = JSON.parse(localStorage.getItem('aceFlowDailyData'));

visitasContainer.innerHTML = '';

visitaCounter = 0;

const rowsToCreate = (data && data.visits.length > INITIAL\_ROWS) ? data.visits.length : INITIAL\_ROWS;

for(let i = 0; i < rowsToCreate; i++) addVisitaCard();

if (!data) return;

agenteInput.value = data.header.agente || agenteInput.value;

bairroInput.value = data.header.bairro || '';

cicloSelect.value = data.header.ciclo || '';

dataInput.value = data.header.data || new Date().toISOString().slice(0, 10);

[agenteInput, bairroInput, cicloSelect, dataInput].forEach(el => { if(el.value) el.classList.add('is-filled'); });

data.visits.forEach((visitData, index) => {

const card = visitasContainer.children[index];

if (!card) return;

Object.keys(visitData).forEach(key => {

const el = card.querySelector(`.${key}`);

if (el) {

if (el.classList.contains('toggle-check')) { if(visitData[key]) el.classList.add('is-checked');}

else { el.value = visitData[key]; }

}

});

updateToggleHTML(card.querySelector('.foco'));

updateToggleHTML(card.querySelector('.tratado'));

if (visitData.isExpanded === "true") {

const header = card.querySelector('.card-header');

header.setAttribute('data-expanded', 'true');

header.nextElementSibling.classList.add('show');

}

updateCardStatus(card);

});

};

const handleSendDataToSheet = async () => {

const btn = document.getElementById('saveRemoteBtn');

btn.disabled = true; btn.innerHTML = 'Enviando...';

const dataToSave = JSON.parse(localStorage.getItem('aceFlowDailyData'));

if (!dataToSave) {

alert("Não há dados para enviar.");

btn.disabled = false; btn.innerHTML = 'Enviar p/ Planilha';

return;

}

const filledVisits = dataToSave.visits.filter(v =>

v.logradouro && v.quarteirao && v.numero && v.lado && v.imovel && v.visita

);

if (filledVisits.length === 0) {

alert("Nenhum imóvel totalmente preenchido para enviar.");

btn.disabled = false; btn.innerHTML = 'Enviar p/ Planilha'; return;

}

// Garante que só os dados completos sejam enviados

dataToSave.visits = filledVisits;

const payload = {

action: 'saveData',

usuario: document.getElementById('username').value,

senha: document.getElementById('password').value,

data: dataToSave

};

try {

const response = await fetch(SCRIPT\_URL, {

method: 'POST', headers: { 'Content-Type': 'text/plain;charset=UTF-8' },

body: JSON.stringify(payload), redirect: 'follow'

});

const result = await response.json();

if (result.status === 'success') {

let history = JSON.parse(localStorage.getItem('aceFlowHistory')) || [];

history.push(payload.data);

localStorage.setItem('aceFlowHistory', JSON.stringify(history));

showModal("Dados enviados com sucesso para a planilha! Deseja limpar a tela para um novo dia?", () => {

localStorage.removeItem('aceFlowDailyData');

location.reload();

});

} else { alert('Erro ao enviar dados:\n\n' + result.message); }

} catch (error) {

console.error('Erro de Rede:', error);

alert('Falha na comunicação com o servidor. Seus dados estão salvos localmente.');

} finally {

btn.disabled = false; btn.innerHTML = 'Enviar p/ Planilha';

}

};

const generatePDF = () => {

const doc = new jsPDF();

const data = JSON.parse(localStorage.getItem('aceFlowDailyData'));

if (!data) return;

// Filtra apenas as visitas completamente preenchidas

const filledVisits = data.visits.filter(v =>

v.logradouro && v.quarteirao && v.numero && v.lado && v.imovel && v.visita

);

if (filledVisits.length === 0) {

alert("Nenhum imóvel totalmente preenchido para gerar o PDF.");

return;

}

const { agente, bairro, ciclo, data: workDate } = data.header;

const dateFormatted = new Date(workDate + 'T12:00:00').toLocaleDateString('pt-BR');

doc.setFontSize(18); doc.setFont('helvetica', 'bold');

doc.text("ACE FLOW - Relatório Diário", 105, 20, { align: 'center' });

doc.autoTable({

startY: 30, theme: 'plain',

body: [['Agente:', agente], ['Data:', dateFormatted], ['Bairro:', bairro], ['Ciclo:', ciclo]]

});

// Recalcula o resumo apenas com os dados filtrados

let summary = { recuperado: 0, fechado: 0, recusado: 0, tratamentoFocal: 0, gramas: 0, depositos: 0, focos: 0 };

let trabalhadosSummary = { total: 0, residencia: 0, comercio: 0, terreno: 0, outro: 0 };

const quarteiroesSet = new Set();

filledVisits.forEach(v => {

if (v.quarteirao && !isNaN(parseInt(v.quarteirao))) quarteiroesSet.add(parseInt(v.quarteirao));

if (v.visita === 'T') {

trabalhadosSummary.total++;

if (v.imovel === 'R') trabalhadosSummary.residencia++;

else if (v.imovel === 'C') trabalhadosSummary.comercio++;

else if (v.imovel === 'TB') trabalhadosSummary.terreno++;

else if (v.imovel === 'O') trabalhadosSummary.outro++;

}

else if (v.visita === 'P') summary.recuperado++;

else if (v.visita === 'F') summary.fechado++;

else if (v.visita === 'R') summary.recusado++;

if(v.trat) summary.tratamentoFocal++;

summary.gramas += parseFloat(v.gram) || 0;

summary.depositos += parseInt(v.dep) || 0;

if(v.foco) summary.focos++;

});

const quarteiroesTrabalhados = quarteiroesSet.size;

let quarteiroesConcluidos = 0;

if (quarteiroesSet.size > 1) {

const maxQuarteirao = Math.max(...quarteiroesSet);

quarteiroesConcluidos = [...quarteiroesSet].filter(q => q < maxQuarteirao).length;

}

// Tabela 1: Detalhes dos Imóveis

const detalhesData = [

['Residências', trabalhadosSummary.residencia],

['Comércios', trabalhadosSummary.comercio],

['Ter. Baldios', trabalhadosSummary.terreno],

['Outros', trabalhadosSummary.outro],

['Total Trabalhados', trabalhadosSummary.total]

];

doc.autoTable({

startY: doc.lastAutoTable.finalY + 10,

head: [['Detalhes dos Imóveis Trabalhados', 'Total']],

body: detalhesData,

headStyles: { fillColor: [0, 0, 0] },

columnStyles: { 0: { fontStyle: 'bold' } },

});

// Tabela 2: Resumo da Produção

const producaoData = [

['Trat. Focal', summary.tratamentoFocal],

['Larvicida', `${summary.gramas.toFixed(1)}g`],

['Dep. Tratados', summary.depositos],

['Total de Focos', summary.focos],

['Recusados', summary.recusado],

['Fechados', summary.fechado],

['Recuperados', summary.recuperado],

['Quart. Trabalhados', quarteiroesTrabalhados],

['Quart. Concluídos', quarteiroesConcluidos]

];

const highlightItems = ['Trat. Focal', 'Larvicida', 'Dep. Tratados', 'Quart. Trabalhados', 'Quart. Concluídos'];

doc.autoTable({

startY: doc.lastAutoTable.finalY + 5,

head: [['Resumo Geral da Produção', 'Total']],

body: producaoData,

headStyles: { fillColor: [0, 0, 0] },

didParseCell: function (data) {

if (data.section === 'body' && highlightItems.includes(data.cell.raw)) {

data.cell.styles.fontStyle = 'bold';

data.cell.styles.textColor = [22, 163, 74]; // Cor --confirm-green

}

}

});

// Tabela 3: Lista de Visitas

const visitData = filledVisits.map((v, i) => [

i + 1, v.logradouro, v.quarteirao, v.numero, v.visita ? v.visita.charAt(0) : '', v.foco ? 'SIM' : ''

]);

doc.autoTable({

head: [['#', 'Logradouro', 'Q.', 'Nº', 'Visita', 'Foco']], body: visitData,

startY: doc.lastAutoTable.finalY + 10, headStyles: { fillColor: [0, 0, 0] }

});

doc.save(`ACE\_FLOW\_${agente.split(' ')[0]}\_${workDate}.pdf`);

};

const updateAutocomplete = (type, value) => {

let list, datalist;

if (type === 'agent') { list = agentList; datalist = agentSuggestions; }

else if (type === 'bairro') { list = bairroList; datalist = bairroSuggestions; }

else { list = streetList; datalist = streetSuggestions; }

const trimmedValue = value.trim();

if (trimmedValue && !list.includes(trimmedValue)) {

list.push(trimmedValue);

datalist.innerHTML = list.map(item => `<option value="${item}"></option>`).join('');

localStorage.setItem(`aceFlow${type}List`, JSON.stringify(list));

}

};

// --- EVENT LISTENERS ---

loginForm.addEventListener('submit', (e) => {

e.preventDefault();

const user = document.getElementById('username').value;

const pass = document.getElementById('password').value;

loginError.textContent = "";

if (!user || !pass) { loginError.textContent = "Por favor, preencha usuário e senha."; return; }

loginBtn.disabled = true; loginBtn.textContent = "Verificando...";

const authPayload = { action: 'authenticate', usuario: user, senha: pass };

fetch(SCRIPT\_URL, {

method: 'POST', headers: { 'Content-Type': 'text/plain;charset=UTF-8' },

body: JSON.stringify(authPayload)

}).then(res => res.json()).then(result => {

if (result.status === 'success') {

loginModal.style.display = 'none';

mainContent.style.display = 'block';

fixedInfoBar.classList.remove('hidden');

agenteInput.value = user;

agenteInput.classList.add('is-filled'); // Add green border

fixedAgentName.textContent = user;

loadData();

updateSummary();

} else { loginError.textContent = result.message || "Usuário ou senha inválidos."; }

}).catch(error => {

console.error("Erro de autenticação:", error);

loginError.textContent = "Erro de conexão. Tente novamente.";

}).finally(() => {

loginBtn.disabled = false; loginBtn.textContent = "Entrar";

});

});

calculateGoalBtn.addEventListener('click', handleGoalCalculation);

document.getElementById('addVisitaBtn').addEventListener('click', addVisitaCard);

document.getElementById('savePdfBtn').addEventListener('click', generatePDF);

document.getElementById('saveRemoteBtn').addEventListener('click', handleSendDataToSheet);

document.getElementById('clearDayBtn').addEventListener('click', () => {

showModal("Tem certeza que deseja limpar os dados do dia? Esta ação não pode ser desfeita.", () => {

localStorage.removeItem('aceFlowDailyData');

location.reload();

});

});

reportBtn.addEventListener('click', () => reportModal.classList.remove('hidden'));

closeReportModalBtn.addEventListener('click', () => reportModal.classList.add('hidden'));

mainContent.addEventListener('input', (e) => {

if (e.target.matches('.form-input, .form-select')) {

const card = e.target.closest('.visita-card');

if (card) updateCardStatus(card);

if(e.target.value) e.target.classList.add('is-filled'); else e.target.classList.remove('is-filled');

if (agenteInput.value) agenteInput.classList.add('is-filled'); // Keep green border

if (dataInput.value) dataInput.classList.add('is-filled'); // Keep green border

updateSummary();

saveData();

}

});

mainContent.addEventListener('change', (e) => {

const target = e.target;

if (target.matches('.lado')) {

const card = target.closest('.visita-card');

if (!card) return;

let valueToReplicate = target.value;

let currentCard = card;

while ((currentCard = currentCard.nextElementSibling) !== null) {

if (currentCard.matches('.visita-card')) {

const selectToUpdate = currentCard.querySelector('.lado');

// Only update if it hasn't been manually selected yet.

if (selectToUpdate && !selectToUpdate.value) {

selectToUpdate.value = valueToReplicate;

if (valueToReplicate) {

selectToUpdate.classList.add('is-filled');

}

updateCardStatus(currentCard);

}

}

}

saveData(); // Save changes after replication

}

});

mainContent.addEventListener('blur', (e) => {

const target = e.target;

if (target.matches('#agente, #bairro, .logradouro') && target.value) {

target.value = formatAndCorrectText(target.value);

}

if (target.id === 'agente') updateAutocomplete('agent', target.value);

if (target.id === 'bairro') updateAutocomplete('bairro', target.value);

if (target.classList.contains('logradouro')) updateAutocomplete('street', target.value);

const card = target.closest('.visita-card');

if (card && (target.matches('.logradouro') || target.matches('.quarteirao'))) {

let valueToReplicate = target.value;

let currentCard = card;

const selector = target.matches('.logradouro') ? '.logradouro' : '.quarteirao';

while ((currentCard = currentCard.nextElementSibling) !== null) {

if (currentCard.matches('.visita-card')) {

const inputToUpdate = currentCard.querySelector(selector);

if (inputToUpdate && !inputToUpdate.value.trim()) {

inputToUpdate.value = valueToReplicate;

if (valueToReplicate) inputToUpdate.classList.add('is-filled');

updateCardStatus(currentCard);

}

}

}

}

saveData();

}, true);

visitasContainer.addEventListener('focusin', (e) => {

const card = e.target.closest('.visita-card');

if(card) {

const logradouro = card.querySelector('.logradouro').value || '...';

const numero = card.querySelector('.numero').value || '...';

fixedPropertyInfo.textContent = `Imóvel: ${logradouro}, ${numero}`;

}

});

visitasContainer.addEventListener('click', (e) => {

const header = e.target.closest('.card-header');

if (header) {

const content = header.nextElementSibling;

const isExpanded = header.getAttribute('data-expanded') === 'true';

header.setAttribute('data-expanded', !isExpanded);

content.classList.toggle('show');

saveData();

return;

}

if (e.target.matches('.toggle-check')) {

e.target.classList.toggle('is-checked');

updateToggleHTML(e.target);

updateSummary();

saveData();

}

});

modalConfirmBtn.addEventListener('click', () => { if (confirmCallback) confirmCallback(); hideModal(); });

modalCancelBtn.addEventListener('click', hideModal);

// --- INICIALIZAÇÃO DO APP ---

const initApp = () => {

agentList = JSON.parse(localStorage.getItem('aceFlowagentList')) || [];

bairroList = JSON.parse(localStorage.getItem('aceFlowbairroList')) || [];

streetList = JSON.parse(localStorage.getItem('aceFlowstreetList')) || [];

agentSuggestions.innerHTML = agentList.map(item => `<option value="${item}"></option>`).join('');

bairroSuggestions.innerHTML = bairroList.map(item => `<option value="${item}"></option>`).join('');

streetSuggestions.innerHTML = streetList.map(item => `<option value="${item}"></option>`).join('');

populateSelects();

dataInput.value = new Date().toISOString().slice(0, 10);

dataInput.classList.add('is-filled'); // Add green border to date by default

loadGoalData();

};

initApp();

});

</script>

</body>

</html>