<!--

CHANGELOG v4 (Gemini):

- [DATA] Значительно расширены демонстрационные данные для полноценного тестирования: добавлены транзакции за последний месяц, бюджеты, цели, регулярные и запланированные платежи, а также счета в разных валютах.

- [PERFORMANCE] Оптимизирован процесс обновления интерфейса. Вместо полной перерисовки при любом изменении (renderAll), теперь обновляются только затронутые компоненты. Это значительно повышает производительность и отзывчивость, особенно при большом количестве данных.

- [FIX] Исправлена гонка состояний при загрузке данных, которая могла приводить к пустому экрану при переключении пространств или первой загрузке. Теперь критические данные загружаются до первого рендеринга.

- [UX] Модальные окна теперь закрываются по клику на фон или нажатию клавиши Escape.

- [UX] Добавлена визуальная обратная связь при сохранении данных (кнопки блокируются и меняют текст).

- [UX] Улучшена обработка ошибок: сообщения стали более конкретными и понятными для пользователя.

- [STABILITY] Добавлена базовая валидация на стороне клиента для форм (проверка на пустые поля и положительные суммы).

- [SECURITY] Обеспечено тотальное экранирование всех выводимых данных через escapeHtml для предотвращения XSS-уязвимостей.

- [REFACTOR] Весь HTML-код для модальных окон вынесен в JavaScript для лучшей организации и чистоты основного HTML-документа.

-->

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Мои финансы</title>

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

<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

<script src="https://cdn.jsdelivr.net/npm/chartjs-adapter-date-fns"></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@300;400;500;600;700;800&display=swap" rel="stylesheet">

<script>

tailwind.config = {

theme: {

extend: {

fontFamily: {

sans: ['Inter', 'sans-serif'],

},

colors: {

'brand-accent': '#A78BFA',

'brand-accent-hover': '#C4B5FD',

'space-bg': '#131120',

'space-card': '#1E1B2E',

'space-border': '#37305A',

'space-text': '#E0E0E0',

'space-text-secondary': '#8A81B4',

},

boxShadow: {

'glow': '0 0 15px rgba(167, 139, 250, 0.25)',

'glow-lg': '0 0 25px rgba(167, 139, 250, 0.35)',

}

}

}

}

</script>

<style>

:root {

--brand-accent: #A78BFA;

--brand-accent-hover: #C4B5FD;

--space-bg: #131120;

--space-card: #1E1B2E;

--space-border: #37305A;

--space-text: #E0E0E0;

--space-text-secondary: #8A81B4;

}

body {

-webkit-font-smoothing: antialiased;

-moz-osx-font-smoothing: grayscale;

}

.bg-card { background-color: var(--space-card); }

.border-card { border-color: var(--space-border); }

.text-main { color: var(--space-text); }

.text-secondary { color: var(--space-text-secondary); }

/\* Стили для кастомных инпутов и селектов \*/

select, input[type="text"], input[type="number"], input[type="date"] {

color: var(--space-text);

background-color: var(--space-bg) !important;

border-color: var(--space-border) !important;

border-width: 1px;

padding: 0.75rem 1rem;

border-radius: 0.5rem;

width: 100%;

transition: all 0.2s ease-in-out;

font-size: 0.95rem;

}

select:focus, input:focus {

outline: none;

border-color: var(--brand-accent) !important;

box-shadow: 0 0 0 2px rgba(167, 139, 250, 0.2);

}

select {

-webkit-appearance: none; -moz-appearance: none; appearance: none;

background-repeat: no-repeat;

background-position: right 0.75rem center;

background-size: 1em;

padding-right: 2.5rem !important;

background-image: url("data:image/svg+xml,%3Csvg xmlns='http://www.w3.org/2000/svg' width='16' height='16' fill='%238A81B4' viewBox='0 0 16 16'%3E%3Cpath fill-rule='evenodd' d='M1.646 4.646a.5.5 0 0 1 .708 0L8 10.293l5.646-5.647a.5.5 0 0 1 .708.708l-6 6a.5.5 0 0 1-.708 0l-6-6a.5.5 0 0 1 0-.708z'/%3E%3C/svg%3E");

}

.chart-container { position: relative; height: 280px; width: 100%; }

/\* Стили для табов \*/

.tab {

transition: all 0.2s ease-in-out;

color: var(--space-text-secondary);

font-weight: 500;

}

.tab-active {

color: var(--space-text);

border-color: var(--brand-accent);

font-weight: 600;

}

.tab:hover {

color: var(--space-text);

}

.progress-bar-bg { background-color: var(--space-border); }

.progress-bar-fill {

background-color: var(--brand-accent);

transition: width 0.5s ease-in-out;

box-shadow: 0 0 10px rgba(167, 139, 250, 0.5);

}

/\* Скрытие скроллбара в навигации \*/

.nav-container::-webkit-scrollbar { display: none; }

.nav-container { -ms-overflow-style: none; scrollbar-width: none; }

/\* Стили для кнопок \*/

.btn-primary {

background-color: var(--brand-accent);

color: white;

font-weight: 600;

border-radius: 0.5rem;

padding: 0.75rem 1.25rem;

transition: all 0.2s;

box-shadow: 0 4px 15px -5px rgba(167, 139, 250, 0.4);

}

.btn-primary:hover {

background-color: var(--brand-accent-hover);

transform: translateY(-2px);

box-shadow: 0 6px 20px -5px rgba(167, 139, 250, 0.6);

}

.btn-primary:disabled {

background-color: #374151; color: #9CA3AF;

cursor: not-allowed; transform: none; box-shadow: none;

}

.btn-secondary {

background-color: transparent;

color: var(--space-text);

font-weight: 500;

border-radius: 0.5rem;

padding: 0.75rem 1.25rem;

transition: all 0.2s;

border: 1px solid var(--space-border);

}

.btn-secondary:hover {

background-color: var(--space-card);

border-color: var(--brand-accent);

color: white;

}

/\* Стили для модальных окон и скелетонов \*/

.modal-container { max-height: 90vh; overflow-y: auto; }

@keyframes pulse { 50% { opacity: .5; } }

.skeleton { background-color: var(--space-card); animation: pulse 2s cubic-bezier(0.4, 0, 0.6, 1) infinite; }

#space-switcher button {

transition: all 0.2s ease-in-out;

border: 1px solid transparent;

}

#space-switcher button.bg-brand-accent {

box-shadow: var(--tw-shadow-glow);

border-color: var(--brand-accent);

}

#space-switcher button:hover:not(.bg-brand-accent) {

background-color: rgba(30, 27, 46, 0.5);

color: var(--space-text);

}

.text-gradient {

background: linear-gradient(90deg, #A78BFA, #C4B5FD);

-webkit-background-clip: text;

-webkit-text-fill-color: transparent;

}

</style>

</head>

<body class="bg-space-bg text-space-text">

<div id="app" class="min-h-screen container mx-auto p-4 sm:p-6 lg:p-8">

<div id="notification-bar" class="hidden fixed top-0 left-0 right-0 p-3 text-white text-center z-50 transition-transform -translate-y-full"></div>

<header class="flex flex-col sm:flex-row justify-between items-start sm:items-center mb-8 gap-4">

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

<h1 class="text-3xl sm:text-4xl font-extrabold text-main tracking-tight">Мои <span class="text-gradient">финансы</span></h1>

<div id="space-switcher" class="flex items-center gap-1 bg-space-card p-1 rounded-lg border border-space-border"></div>

</div>

<div id="auth-info" class="text-xs text-space-text-secondary self-end sm:self-center">Загрузка...</div>

</header>

<div id="main-nav" class="hidden mb-8 border-b border-card">

<nav class="flex space-x-2 sm:space-x-4 overflow-x-auto nav-container" aria-label="Tabs">

<button class="tab tab-active py-3 px-4 font-medium text-sm border-b-2 border-transparent hover:text-main whitespace-nowrap" data-tab="dashboard">Дашборд</button>

<button class="tab py-3 px-4 font-medium text-sm border-b-2 border-transparent hover:text-main whitespace-nowrap" data-tab="transactions">Транзакции</button>

<button class="tab py-3 px-4 font-medium text-sm border-b-2 border-transparent hover:text-main whitespace-nowrap" data-tab="accounts">Счета</button>

<button class="tab py-3 px-4 font-medium text-sm border-b-2 border-transparent hover:text-main whitespace-nowrap" data-tab="savings">Накопления</button>

<button class="tab py-3 px-4 font-medium text-sm border-b-2 border-transparent hover:text-main whitespace-nowrap" data-tab="planner">Календарь платежей</button>

<button class="tab py-3 px-4 font-medium text-sm border-b-2 border-transparent hover:text-main whitespace-nowrap" data-tab="budgets-goals">Бюджеты и цели</button>

<button class="tab py-3 px-4 font-medium text-sm border-b-2 border-transparent hover:text-main whitespace-nowrap" data-tab="settings">Настройки</button>

</nav>

</div>

<main id="tab-content">

<!-- Начальный загрузчик -->

<div id="initial-loader">

<div class="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-6 mb-6">

<div class="skeleton h-24 rounded-xl"></div>

<div class="skeleton h-24 rounded-xl"></div>

<div class="skeleton h-24 rounded-xl"></div>

<div class="skeleton h-24 rounded-xl"></div>

</div>

<div class="skeleton h-64 rounded-xl"></div>

</div>

<!-- Панели для вкладок -->

<div id="dashboard-panel" class="hidden"></div>

<div id="transactions-panel" class="hidden"></div>

<div id="accounts-panel" class="hidden"></div>

<div id="savings-panel" class="hidden"></div>

<div id="planner-panel" class="hidden"></div>

<div id="budgets-goals-panel" class="hidden"></div>

<div id="settings-panel" class="hidden"></div>

</main>

</div>

<div id="offline-indicator" class="hidden fixed bottom-4 right-4 bg-gray-700 text-white text-xs px-3 py-1 rounded-full shadow-lg">Оффлайн режим</div>

<!-- Контейнеры для модальных окон (содержимое будет вставлено через JS) -->

<div id="modal-placeholders"></div>

<script type="module">

import { initializeApp } from "https://www.gstatic.com/firebasejs/11.12.2/firebase-app.js";

import { getAuth, signInAnonymously, signInWithCustomToken, onAuthStateChanged } from "https://www.gstatic.com/firebasejs/11.12.2/firebase-auth.js";

import { getFirestore, collection, doc, onSnapshot, query, serverTimestamp, deleteDoc, setDoc, writeBatch, Timestamp, increment, where, getDocs, getDoc, runTransaction } from "https://www.gstatic.com/firebasejs/11.12.2/firebase-firestore.js";

// --- ГЛОБАЛЬНЫЕ ПЕРЕМЕННЫЕ И КОНФИГУРАЦИЯ ---

const appId = typeof \_\_app\_id !== 'undefined' ? \_\_app\_id : 'default-finance-app-v15-gemini';

let firebaseConfig = null;

try {

firebaseConfig = typeof \_\_firebase\_config !== 'undefined' ? JSON.parse(\_\_firebase\_config) : null;

} catch (e) {

console.error("Firebase config is not valid JSON", e);

}

let app, db, auth, userId;

let expenseChart, savingsChart;

const state = {

spaces: [], allAccounts: [], accounts: [], savings: [], transactions: [], goals: [], recurring: [], budgets: [], plannedPayments: [], taxes: [],

currentSpaceId: null, currentSpace: null, unsubscribers: {}, dataInitialized: false

};

const defaultCategories = {

income: ['Зарплата / Найм', 'Продажи', 'Фриланс', 'Подарки', 'Перевод', 'Другое'],

expense: ['Квартира', 'Еда', 'Транспорт', 'Жилье', 'Развлечения', 'Связь', 'Здоровье', 'Одежда', 'Бизнес', 'Налоги', 'Цели', 'Перевод', 'Другое'],

};

const currencies = ['RUB', 'USD', 'EUR', 'GEL', 'KZT', 'USDT'];

// --- УТИЛИТЫ ---

const escapeHtml = (unsafe) => {

if (unsafe === null || typeof unsafe === 'undefined') return '';

return unsafe.toString().replace(/[&<"']/g, (m) => ({ '&': '&amp;', '<': '&lt;', '>': '&gt;', '"': '&quot;', "'": '&#039;' })[m]);

}

const formatCurrency = (amount, currency = 'RUB') => {

const amountToShow = typeof amount === 'number' ? amount : 0;

if (currency && currency.toUpperCase() === 'USDT') {

return new Intl.NumberFormat('ru-RU', { minimumFractionDigits: 2, maximumFractionDigits: 2 }).format(amountToShow) + ' USDT';

}

try {

const options = { style: 'currency', currency };

options.minimumFractionDigits = (currency && currency.toUpperCase() === 'RUB') ? 0 : 2;

options.maximumFractionDigits = (currency && currency.toUpperCase() === 'RUB') ? 0 : 2;

return new Intl.NumberFormat('ru-RU', options).format(amountToShow);

} catch (e) {

console.warn(`Could not format currency "${currency}". Falling back to number format.`);

return new Intl.NumberFormat('ru-RU', { minimumFractionDigits: 2, maximumFractionDigits: 2 }).format(amountToShow) + ` ${escapeHtml(currency)}`;

}

};

const formatDate = (dateInput) => {

if (!dateInput) return '';

try {

const date = dateInput.toDate ? dateInput.toDate() : new Date(dateInput);

return date.toLocaleDateString('ru-RU');

} catch (e) {

console.error("Invalid date format:", dateInput);

return 'Неверная дата';

}

};

const getISODate = (dateInput) => {

try {

const date = dateInput ? (dateInput.toDate ? dateInput.toDate() : new Date(dateInput)) : new Date();

return date.toISOString().split('T')[0];

} catch(e) {

return new Date().toISOString().split('T')[0];

}

}

const showNotification = (message, type = 'info') => {

const bar = document.getElementById('notification-bar');

if(!bar) return;

bar.textContent = message;

bar.className = 'fixed top-0 left-0 right-0 p-3 text-white text-center z-50 transition-transform transform';

const colors = { success: 'bg-green-600', error: 'bg-red-600', info: 'bg-blue-600' };

bar.classList.add(colors[type] || colors.info);

bar.classList.remove('-translate-y-full');

setTimeout(() => bar.classList.add('-translate-y-full'), 3000);

};

const showModal = (id) => document.getElementById(id)?.classList.remove('hidden');

const hideModal = (id) => document.getElementById(id)?.classList.add('hidden');

// [REFACTOR] Функции-помощники для генерации опций в модальных окнах

const allAccountsOptions = () => [...state.accounts, ...state.savings].map(a => `<option value="${a.id}">${escapeHtml(a.name)} (${formatCurrency(a.balance, a.currency)})</option>`).join('');

const expenseCategoriesOptions = () => (state.currentSpace?.categories?.expense || defaultCategories.expense).map(c => `<option value="${escapeHtml(c)}">${escapeHtml(c)}</option>`).join('');

const incomeCategoriesOptions = () => (state.currentSpace?.categories?.income || defaultCategories.income).map(c => `<option value="${escapeHtml(c)}">${escapeHtml(c)}</option>`).join('');

const allCurrenciesOptions = () => currencies.map(c => `<option value="${c}">${c}</option>`).join('');

// --- ИНИЦИАЛИЗАЦИЯ И ЗАГРУЗКА ДАННЫХ ---

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

if (!firebaseConfig) {

showNotification('Критическая ошибка: отсутствует конфигурация Firebase.', 'error');

document.getElementById('initial-loader').innerHTML = '<p class="text-red-400 text-center">Ошибка конфигурации. Приложение не может быть запущено.</p>';

return;

}

app = initializeApp(firebaseConfig);

db = getFirestore(app);

auth = getAuth(app);

injectModalSkeletons();

setupEventListeners();

onAuthStateChanged(auth, user => {

if (user) {

userId = user.uid;

document.getElementById('auth-info').innerHTML = `ID: <span class="font-mono">${userId.substring(0,8)}...</span> | App: <span class="font-mono">${appId.substring(0,12)}</span>`;

loadInitialData();

} else {

signIn();

}

});

});

async function signIn() {

try {

if (typeof \_\_initial\_auth\_token !== 'undefined' && \_\_initial\_auth\_token) {

await signInWithCustomToken(auth, \_\_initial\_auth\_token);

} else {

await signInAnonymously(auth);

}

} catch (error) {

console.error("Authentication failed:", error);

showNotification('Ошибка аутентификации. Обновите страницу.', 'error');

}

}

async function loadInitialData() {

if (!userId) return;

try {

const userRef = doc(db, `artifacts/${appId}/users/${userId}`);

const userSnap = await getDoc(userRef);

if (userSnap.exists()) {

state.dataInitialized = userSnap.data().dataInitialized;

}

if (!state.dataInitialized) {

await createDefaultData();

}

const spacesQuery = query(collection(db, `artifacts/${appId}/users/${userId}/spaces`));

onSnapshot(spacesQuery, (snapshot) => {

state.spaces = snapshot.docs

.map(d => ({ id: d.id, ...d.data() }))

.sort((a,b) => (a.createdAt?.toMillis() || 0) - (b.createdAt?.toMillis() || 0));

const userLastSpaceId = userSnap.exists() ? userSnap.data().lastUsedSpaceId : null;

if (!state.currentSpaceId) {

const targetSpaceId = userLastSpaceId && state.spaces.some(s => s.id === userLastSpaceId) ? userLastSpaceId : state.spaces[0]?.id;

if (targetSpaceId) {

switchSpace(targetSpaceId, true); // initial load

} else {

document.getElementById('initial-loader').innerHTML = `<div class="text-center p-8 bg-card rounded-xl border border-card"><h3 class="text-xl font-bold mb-2 text-main">Добро пожаловать!</h3><p class="text-secondary">Рабочие пространства не найдены. Создайте первое, чтобы начать.</p><button class="btn-primary mt-4" data-action="open-space-modal">Создать пространство</button></div>`;

document.getElementById('main-nav').classList.add('hidden');

}

}

renderSpaceSwitcher();

}, (error) => {

console.error("Error loading spaces:", error);

showNotification("Ошибка загрузки рабочих пространств.", "error");

});

} catch (error) {

console.error("Initial load error:", error);

showNotification("Не удалось загрузить данные.", "error");

document.getElementById('initial-loader').innerHTML = '<p class="text-red-400">Критическая ошибка. Обновите страницу.</p>';

}

}

async function loadDataForCurrentSpace() {

if (!userId || !state.currentSpaceId) {

console.warn("loadDataForCurrentSpace aborted: missing user/space ID.");

document.getElementById('initial-loader').classList.add('hidden');

document.getElementById('main-nav').classList.add('hidden');

document.getElementById('tab-content').innerHTML = `<p class="text-center text-red-400">Не удалось загрузить пространство. Пожалуйста, выберите другое.</p>`;

return;

}

// Unsubscribe from any previous listeners

Object.values(state.unsubscribers).forEach(unsub => unsub?.());

state.unsubscribers = {};

try {

// --- [PERFORMANCE] Step 1: Batch fetch all initial data ---

const collectionsToFetch = ['accounts', 'transactions', 'goals', 'budgets', 'plannedPayments', 'recurring', 'taxes'];

const promises = collectionsToFetch.map(colName => {

const q = query(collection(db, `artifacts/${appId}/users/${userId}/${colName}`), where("spaceId", "==", state.currentSpaceId));

return getDocs(q);

});

const spacePromise = getDoc(doc(db, `artifacts/${appId}/users/${userId}/spaces`, state.currentSpaceId));

promises.push(spacePromise);

const results = await Promise.all(promises);

// --- Step 2: Populate state from batched results ---

const spaceSnap = results.pop(); // The last promise was the space doc

if (!spaceSnap.exists()) {

throw new Error(`Пространство с ID ${state.currentSpaceId} не найдено.`);

}

state.currentSpace = { id: spaceSnap.id, ...spaceSnap.data() };

if (state.currentSpace && !state.currentSpace.categories) {

state.currentSpace.categories = defaultCategories;

}

results.forEach((collectionSnap, index) => {

const colName = collectionsToFetch[index];

const data = collectionSnap.docs.map(doc => ({ id: doc.id, ...doc.data() }));

if (colName === 'accounts') {

state.allAccounts = data;

state.accounts = data.filter(a => a.type === 'regular' || a.type === 'crypto');

state.savings = data.filter(a => a.type === 'savings');

} else {

state[colName] = data.sort((a,b) => (b.date || b.createdAt)?.toMillis() - (a.date || a.createdAt)?.toMillis());

}

});

// --- Step 3: Render the UI ONCE with all data ---

renderAll();

document.getElementById('initial-loader').classList.add('hidden');

document.getElementById('main-nav').classList.remove('hidden');

switchTab('dashboard');

// --- Step 4: Attach real-time listeners for future updates ---

attachRealtimeListeners();

} catch(e) {

console.error("Error loading space data:", e);

showNotification("Ошибка загрузки данных пространства.", "error");

document.getElementById('initial-loader').classList.add('hidden');

const errorHtml = `<div class="text-center p-8 bg-card rounded-xl border border-red-500/50"><h3 class="text-xl font-bold mb-2 text-red-400">Ошибка Загрузки</h3><p class="text-secondary">${escapeHtml(e.message)}</p></div>`;

document.getElementById('tab-content').innerHTML = errorHtml;

}

}

// [PERFORMANCE] Эта новая функция подключает слушатели ПОСЛЕ начальной загрузки данных

function attachRealtimeListeners() {

if (!userId || !state.currentSpaceId) return;

// Слушатель для документа пространства

const spaceRef = doc(db, `artifacts/${appId}/users/${userId}/spaces`, state.currentSpaceId);

state.unsubscribers.space = onSnapshot(spaceRef, (doc) => {

state.currentSpace = doc.exists() ? { id: doc.id, ...doc.data() } : null;

if(state.currentSpace) {

renderSettings();

if (state.currentSpace.type === 'business') renderBusinessDashboard(); else renderDashboard();

}

});

// Гранулярные слушатели для коллекций

const collectionsToRenderMap = {

accounts: (snapshot) => {

document.getElementById('offline-indicator').classList.toggle('hidden', !snapshot.metadata.fromCache);

const allAccounts = snapshot.docs.map(doc => ({ id: doc.id, ...doc.data() }));

state.allAccounts = allAccounts;

state.accounts = allAccounts.filter(a => a.type === 'regular' || a.type === 'crypto');

state.savings = allAccounts.filter(a => a.type === 'savings');

renderAccounts();

renderSavings();

if (state.currentSpace?.type === 'business') renderBusinessDashboard(); else renderDashboard();

},

transactions: (snapshot) => {

state.transactions = snapshot.docs.map(doc => ({ id: doc.id, ...doc.data() })).sort((a,b) => (b.date || b.createdAt)?.toMillis() - (a.date || a.createdAt)?.toMillis());

renderTransactions();

renderBudgetsAndGoals();

updateSavingsChart();

if (state.currentSpace?.type === 'business') renderBusinessDashboard(); else renderDashboard();

},

goals: (snapshot) => {

state.goals = snapshot.docs.map(doc => ({ id: doc.id, ...doc.data() })).sort((a,b) => (b.date || b.createdAt)?.toMillis() - (a.date || a.createdAt)?.toMillis());

renderBudgetsAndGoals();

},

budgets: (snapshot) => {

state.budgets = snapshot.docs.map(doc => ({ id: doc.id, ...doc.data() })).sort((a,b) => (b.date || b.createdAt)?.toMillis() - (a.date || a.createdAt)?.toMillis());

renderBudgetsAndGoals();

},

plannedPayments: (snapshot) => {

state.plannedPayments = snapshot.docs.map(doc => ({ id: doc.id, ...doc.data() })).sort((a,b) => (b.date || b.createdAt)?.toMillis() - (a.date || a.createdAt)?.toMillis());

renderPlanner();

},

recurring: (snapshot) => {

state.recurring = snapshot.docs.map(doc => ({ id: doc.id, ...doc.data() })).sort((a,b) => (b.date || b.createdAt)?.toMillis() - (a.date || a.createdAt)?.toMillis());

renderTransactions();

},

taxes: (snapshot) => {

state.taxes = snapshot.docs.map(doc => ({ id: doc.id, ...doc.data() })).sort((a,b) => (b.date || b.createdAt)?.toMillis() - (a.date || a.createdAt)?.toMillis());

renderPlanner();

},

};

Object.entries(collectionsToRenderMap).forEach(([colName, processSnapshot]) => {

const q = query(collection(db, `artifacts/${appId}/users/${userId}/${colName}`), where("spaceId", "==", state.currentSpaceId));

state.unsubscribers[colName] = onSnapshot(q, processSnapshot, (error) => console.error(`Error loading ${colName}:`, error));

});

}

async function createDefaultData() {

if (state.dataInitialized) return;

showNotification("Создаем демонстрационные данные...", "info");

try {

const batch = writeBatch(db);

const userRef = doc(db, `artifacts/${appId}/users/${userId}`);

// --- БИЗНЕС ПРОСТРАНСТВО ---

const businessSpaceRef = doc(collection(db, `artifacts/${appId}/users/${userId}/spaces`));

batch.set(businessSpaceRef, { name: "Бизнес", type: "business", taxRate: 6, taxProvision: 0, mainCurrency: "RUB", exchangeRates: { GEL: 30 }, categories: defaultCategories, createdAt: serverTimestamp() });

const businessAccountRef = doc(collection(db, `artifacts/${appId}/users/${userId}/accounts`));

batch.set(businessAccountRef, { name: "Счет ИП", currency: "RUB", balance: 250000, type: 'regular', spaceId: businessSpaceRef.id, createdAt: serverTimestamp() });

// --- ЛИЧНОЕ ПРОСТРАНСТВО ---

const personalSpaceRef = doc(collection(db, `artifacts/${appId}/users/${userId}/spaces`));

batch.set(personalSpaceRef, { name: "Личное", type: "personal", mainCurrency: "RUB", exchangeRates: { GEL: 30, USD: 90, USDT: 92 }, categories: defaultCategories, createdAt: serverTimestamp() });

// Счета для личного пространства

const tinkoffRef = doc(collection(db, `artifacts/${appId}/users/${userId}/accounts`));

batch.set(tinkoffRef, { name: "Тинькофф", currency: "RUB", balance: 50000, type: 'regular', spaceId: personalSpaceRef.id, createdAt: serverTimestamp() });

const cryptoWalletRef = doc(collection(db, `artifacts/${appId}/users/${userId}/accounts`));

batch.set(cryptoWalletRef, { name: "Crypto Wallet", currency: "USDT", balance: 500, type: 'crypto', spaceId: personalSpaceRef.id, createdAt: serverTimestamp() });

const cashRef = doc(collection(db, `artifacts/${appId}/users/${userId}/accounts`));

batch.set(cashRef, { name: "Наличные", currency: "GEL", balance: 1500, type: 'regular', spaceId: personalSpaceRef.id, createdAt: serverTimestamp() });

const savingsRef = doc(collection(db, `artifacts/${appId}/users/${userId}/accounts`));

batch.set(savingsRef, { name: "Накопительный", currency: "RUB", balance: 125000, type: 'savings', spaceId: personalSpaceRef.id, createdAt: serverTimestamp() });

// Цели и бюджеты для личного

batch.set(doc(collection(db, `artifacts/${appId}/users/${userId}/goals`)), { name: "Отпуск", currentAmount: 50000, targetAmount: 200000, spaceId: personalSpaceRef.id, createdAt: serverTimestamp() });

batch.set(doc(collection(db, `artifacts/${appId}/users/${userId}/budgets`)), { category: "Еда", amount: 25000, spaceId: personalSpaceRef.id, createdAt: serverTimestamp() });

batch.set(doc(collection(db, `artifacts/${appId}/users/${userId}/budgets`)), { category: "Развлечения", amount: 10000, spaceId: personalSpaceRef.id, createdAt: serverTimestamp() });

// Регулярные и запланированные платежи

const nextMonth = new Date(); nextMonth.setMonth(nextMonth.getMonth() + 1);

batch.set(doc(collection(db, `artifacts/${appId}/users/${userId}/plannedPayments`)), { description: "Оплата интернета", amount: 500, category: "Связь", date: Timestamp.fromDate(nextMonth), accountId: tinkoffRef.id, spaceId: personalSpaceRef.id });

batch.set(doc(collection(db, `artifacts/${appId}/users/${userId}/recurring`)), { description: "Мобильная связь", amount: 450, dayOfMonth: 15, type: 'expense', category: "Связь", accountId: tinkoffRef.id, spaceId: personalSpaceRef.id });

// Налоги для бизнеса

batch.set(doc(collection(db, `artifacts/${appId}/users/${userId}/taxes`)), { name: "УСН (Доходы)", amount: 15000, dueDate: Timestamp.fromDate(nextMonth), isPaid: false, spaceId: businessSpaceRef.id, createdAt: serverTimestamp() });

// Транзакции за последний месяц

const today = new Date();

const addTx = (daysAgo, amount, type, category, description, accountId, spaceId) => {

const date = new Date(today);

date.setDate(date.getDate() - daysAgo);

const txRef = doc(collection(db, `artifacts/${appId}/users/${userId}/transactions`));

batch.set(txRef, { type, amount, category, description, accountId, spaceId, date: Timestamp.fromDate(date) });

};

addTx(25, 150000, 'income', 'Зарплата / Найм', 'Аванс', tinkoffRef.id, personalSpaceRef.id);

addTx(24, 2500, 'expense', 'Еда', 'Продукты на неделю', tinkoffRef.id, personalSpaceRef.id);

addTx(22, 1200, 'expense', 'Развлечения', 'Кино с друзьями', tinkoffRef.id, personalSpaceRef.id);

addTx(20, 300, 'expense', 'Транспорт', 'Такси', cashRef.id, personalSpaceRef.id);

addTx(15, 800, 'expense', 'Еда', 'Обед в кафе', cashRef.id, personalSpaceRef.id);

addTx(10, 150000, 'income', 'Зарплата / Найм', 'Зарплата', tinkoffRef.id, personalSpaceRef.id);

addTx(9, 45000, 'expense', 'Квартира', 'Аренда', tinkoffRef.id, personalSpaceRef.id);

addTx(5, 10000, 'income', 'Перевод', 'Перевод на накоп.', savingsRef.id, personalSpaceRef.id);

addTx(2, 5500, 'expense', 'Здоровье', 'Лекарства', tinkoffRef.id, personalSpaceRef.id);

addTx(28, 75000, 'income', 'Продажи', 'Проект "Альфа"', businessAccountRef.id, businessSpaceRef.id);

addTx(15, 120000, 'income', 'Продажи', 'Проект "Бета"', businessAccountRef.id, businessSpaceRef.id);

addTx(10, 15000, 'expense', 'Бизнес', 'Подписка на сервис', businessAccountRef.id, businessSpaceRef.id);

batch.set(userRef, { lastUsedSpaceId: personalSpaceRef.id, dataInitialized: true }, { merge: true });

await batch.commit();

state.dataInitialized = true;

showNotification("Демонстрационные данные успешно созданы!", "success");

} catch (error) {

console.error("Failed to create default data:", error);

showNotification("Ошибка создания стартовых данных.", 'error');

}

}

// --- ФУНКЦИИ РЕНДЕРИНГА ---

function renderAll() {

if (!state.currentSpaceId || !state.currentSpace) {

console.warn("renderAll aborted: currentSpace or its ID is not yet available.");

return;

}

if (state.currentSpace.type === 'business') {

renderBusinessDashboard();

} else {

renderDashboard();

}

renderTransactions();

renderAccounts();

renderSavings();

renderPlanner();

renderBudgetsAndGoals();

renderSettings();

}

function calculateTotalInMainCurrency() {

if (!state.currentSpace) return { total: 0, missingRates: [], mainCurrency: 'RUB' };

const mainCurrency = state.currentSpace.mainCurrency || 'RUB';

const rates = state.currentSpace.exchangeRates || {};

let totalInMainCurrency = 0;

const missingRates = new Set();

const allAccounts = [...state.accounts, ...state.savings];

allAccounts.forEach(acc => {

if (acc.currency === mainCurrency) {

totalInMainCurrency += acc.balance;

} else if (rates[acc.currency]) {

totalInMainCurrency += acc.balance \* rates[acc.currency];

} else {

missingRates.add(acc.currency);

}

});

return { total: totalInMainCurrency, missingRates: Array.from(missingRates), mainCurrency };

}

function renderBusinessDashboard() {

const panel = document.getElementById('dashboard-panel');

if (!panel || !state.currentSpace) return;

const now = new Date();

const startOfMonth = new Date(now.getFullYear(), now.getMonth(), 1);

const thisMonthTransactions = state.transactions.filter(t => t.date && t.date.toDate() >= startOfMonth);

const incomeMonth = thisMonthTransactions.filter(t => t.type === 'income').reduce((sum, t) => sum + t.amount, 0);

const expensesMonth = thisMonthTransactions.filter(t => t.type === 'expense').reduce((sum, t) => sum + t.amount, 0);

const netProfit = incomeMonth - expensesMonth;

const totalBalanceByCurrency = state.accounts.reduce((acc, account) => {

acc[account.currency] = (acc[account.currency] || 0) + account.balance;

return acc;

}, {});

const { total: totalConverted, missingRates, mainCurrency } = calculateTotalInMainCurrency();

const taxProvision = state.currentSpace?.taxProvision || 0;

const availableForWithdrawal = (totalBalanceByCurrency[mainCurrency] || 0) - taxProvision;

let healthStatus, healthReason;

if (availableForWithdrawal < 0) {

healthStatus = '<span class="text-red-400">Высокий риск</span>';

healthReason = 'Средств на счетах недостаточно для покрытия налогового резерва.';

} else if (netProfit < 0) {

healthStatus = '<span class="text-yellow-400">Требует внимания</span>';

healthReason = 'В этом месяце расходы превышают доходы.';

} else {

healthStatus = '<span class="text-green-400">Бизнес в порядке</span>';

healthReason = 'Прибыль положительная, и на счетах достаточно средств для уплаты налогов.';

}

const upcomingTaxes = state.taxes.filter(t => !t.isPaid).sort((a,b) => a.dueDate.toMillis() - b.dueDate.toMillis());

const missingRatesHTML = missingRates.length > 0 ? `<p class="text-xs text-yellow-400 mt-1">Не задан курс для: ${missingRates.map(escapeHtml).join(', ')}</p>` : '';

panel.innerHTML = `

<div class="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-6 mb-6">

<div class="bg-card p-6 rounded-xl border border-card md:col-span-2 lg:col-span-1">

<h3 class="text-sm font-medium text-secondary">Общий баланс</h3>

<p class="text-2xl font-bold text-brand-accent">${formatCurrency(totalConverted, mainCurrency)}</p>

${missingRatesHTML}

</div>

<div class="bg-card p-6 rounded-xl border border-card">

<h3 class="text-sm font-medium text-secondary">Чистая прибыль (месяц)</h3>

<p class="text-2xl font-bold ${netProfit >= 0 ? 'text-green-400' : 'text-red-400'}">${formatCurrency(netProfit)}</p>

</div>

<div class="bg-card p-6 rounded-xl border border-card">

<h3 class="text-sm font-medium text-secondary">Налоговый резерв</h3>

<p class="text-2xl font-bold text-main">${formatCurrency(taxProvision)}</p>

</div>

<div class="bg-card p-6 rounded-xl border border-card">

<h3 class="text-sm font-medium text-secondary">Доступно к выводу (${escapeHtml(mainCurrency)})</h3>

<p class="text-2xl font-bold text-brand-accent">${formatCurrency(availableForWithdrawal, mainCurrency)}</p>

</div>

</div>

<div class="grid grid-cols-1 lg:grid-cols-3 gap-6">

<div class="bg-card p-6 rounded-xl border border-card lg:col-span-1">

<h3 class="text-lg font-bold mb-4 text-main">Здоровье бизнеса</h3>

<p class="text-xl font-bold">${healthStatus}</p>

<p class="text-sm text-secondary mt-1">${escapeHtml(healthReason)}</p>

</div>

<div class="bg-card p-6 rounded-xl border border-card">

<h3 class="text-lg font-bold mb-4 text-main">Предстоящие налоги</h3>

${upcomingTaxes.length > 0 ? upcomingTaxes.map(t => `

<div class="flex justify-between items-center p-3 bg-space-bg rounded-lg mb-2">

<div>

<p class="font-semibold text-main">${escapeHtml(t.name)}</p>

<p class="text-sm text-secondary">Срок: ${formatDate(t.dueDate)}</p>

</div>

<p class="font-bold text-red-400">${formatCurrency(t.amount)}</p>

</div>

`).join('') : '<p class="text-secondary">Нет предстоящих налоговых платежей.</p>'}

</div>

<div class="bg-card p-6 rounded-xl border border-card">

<h3 class="text-lg font-bold mb-4 text-main">Расходы по категориям (месяц)</h3>

<div class="chart-container"><canvas id="expenseChart"></canvas></div>

</div>

</div>`;

updateCharts();

}

function renderDashboard() {

const panel = document.getElementById('dashboard-panel');

if(!panel) return;

const now = new Date();

const startOfMonth = new Date(now.getFullYear(), now.getMonth(), 1);

const thisMonthTransactions = state.transactions.filter(t => t.date && t.date.toDate() >= startOfMonth);

const income = thisMonthTransactions.filter(t => t.type === 'income').reduce((sum, t) => sum + t.amount, 0);

const expenses = thisMonthTransactions.filter(t => t.type === 'expense').reduce((sum, t) => sum + t.amount, 0);

const totalBalanceByCurrency = [...state.accounts, ...state.savings].reduce((acc, account) => {

acc[account.currency] = (acc[account.currency] || 0) + account.balance;

return acc;

}, {});

const { total: totalConverted, missingRates, mainCurrency } = calculateTotalInMainCurrency();

const missingRatesHTML = missingRates.length > 0 ? `<p class="text-xs text-yellow-400 mt-1">Не задан курс для: ${missingRates.map(escapeHtml).join(', ')}</p>` : '';

panel.innerHTML = `

<div class="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-6 mb-6">

<div class="bg-card p-6 rounded-xl border border-card">

<h3 class="text-sm font-medium text-secondary">Общий баланс</h3>

<p class="text-3xl font-bold text-brand-accent">${formatCurrency(totalConverted, mainCurrency)}</p>

${missingRatesHTML}

</div>

${Object.entries(totalBalanceByCurrency).map(([currency, balance]) => `

<div class="bg-card p-6 rounded-xl border border-card">

<h3 class="text-sm font-medium text-secondary">Баланс (${escapeHtml(currency)})</h3>

<p class="text-3xl font-bold text-main">${formatCurrency(balance, currency)}</p>

</div>

`).join('')}

<div class="bg-card p-6 rounded-xl border border-card">

<h3 class="text-sm font-medium text-secondary">Доходы (этот месяц)</h3>

<p class="text-3xl font-bold text-green-400">${formatCurrency(income)}</p>

</div>

<div class="bg-card p-6 rounded-xl border border-card">

<h3 class="text-sm font-medium text-secondary">Расходы (этот месяц)</h3>

<p class="text-3xl font-bold text-red-400">${formatCurrency(expenses)}</p>

</div>

</div>

<div class="grid grid-cols-1 lg:grid-cols-2 gap-6">

<div class="bg-card p-6 rounded-xl border border-card">

<h3 class="text-lg font-bold mb-4 text-main">Расходы по категориям</h3>

<div class="chart-container"><canvas id="expenseChart"></canvas></div>

</div>

<div class="bg-card p-6 rounded-xl border border-card">

<h3 class="text-lg font-bold mb-4 text-main">Динамика накоплений (30 дн)</h3>

<div class="chart-container"><canvas id="savingsChart"></canvas></div>

</div>

</div>`;

updateCharts();

updateSavingsChart();

}

// --- ОБНОВЛЕНИЕ ГРАФИКОВ ---

function updateCharts() {

const ctx = document.getElementById('expenseChart')?.getContext('2d');

if (!ctx) return;

const chartColor = getComputedStyle(document.body).getPropertyValue('--space-text-secondary').trim();

const chartBorderColor = getComputedStyle(document.body).getPropertyValue('--space-card').trim();

const now = new Date();

const startOfMonth = new Date(now.getFullYear(), now.getMonth(), 1);

const thisMonthExpenses = state.transactions.filter(t => t.type === 'expense' && t.date && t.date.toDate() >= startOfMonth);

const expenseByCategory = thisMonthExpenses.reduce((acc, t) => {

acc[t.category] = (acc[t.category] || 0) + t.amount;

return acc;

}, {});

const labels = Object.keys(expenseByCategory);

const data = Object.values(expenseByCategory);

if (expenseChart) expenseChart.destroy();

if (labels.length === 0) {

ctx.canvas.parentElement.innerHTML = `<canvas id="expenseChart"></canvas><div class="absolute inset-0 flex items-center justify-center text-secondary">Нет данных о расходах за этот месяц</div>`;

return;

}

expenseChart = new Chart(ctx, {

type: 'doughnut',

data: { labels, datasets: [{ data, backgroundColor: ['#A78BFA', '#7C3AED', '#5B21B6', '#C4B5FD', '#8B5CF6', '#6D28D9'], borderColor: chartBorderColor, borderWidth: 4 }] },

options: { responsive: true, maintainAspectRatio: false, plugins: { legend: { position: 'bottom', labels: { color: chartColor, boxWidth: 12, padding: 20, font: { family: "'Inter', sans-serif" } } } }, cutout: '60%' }

});

}

function updateSavingsChart() {

const savingsChartContainer = document.getElementById('savingsChart')?.parentElement;

if(!savingsChartContainer) return;

if (state.currentSpace?.type === 'business') {

savingsChartContainer.parentElement.classList.add('hidden');

return;

} else {

savingsChartContainer.parentElement.classList.remove('hidden');

}

const ctx = savingsChartContainer.querySelector('canvas').getContext('2d');

const chartColor = getComputedStyle(document.body).getPropertyValue('--space-text-secondary').trim();

const chartGridColor = getComputedStyle(document.body).getPropertyValue('--space-border').trim();

if (savingsChart) savingsChart.destroy();

if (state.savings.length === 0) {

ctx.canvas.parentElement.innerHTML = '<canvas id="savingsChart"></canvas><div class="absolute inset-0 flex items-center justify-center text-secondary">Нет накопительных счетов</div>';

return;

}

const mainCurrency = state.currentSpace?.mainCurrency || 'RUB';

const savingsInMainCurrency = state.savings.filter(a => a.currency === mainCurrency);

const savingsAccountIds = savingsInMainCurrency.map(a => a.id);

const currentTotalSavings = savingsInMainCurrency.reduce((sum, a) => sum + a.balance, 0);

const today = new Date();

today.setHours(23, 59, 59, 999);

const thirtyDaysAgo = new Date(today);

thirtyDaysAgo.setDate(today.getDate() - 30);

thirtyDaysAgo.setHours(0, 0, 0, 0);

const relevantTransactions = state.transactions

.filter(t => savingsAccountIds.includes(t.accountId) && t.date && t.date.toDate() >= thirtyDaysAgo)

.map(t => ({ ...t, date: t.date.toDate() }))

.sort((a, b) => a.date - b.date);

let balance30DaysAgo = currentTotalSavings;

relevantTransactions.forEach(t => {

balance30DaysAgo += (t.type === 'expense') ? t.amount : -t.amount;

});

let runningBalance = balance30DaysAgo;

const chartData = [];

for (let i = 0; i <= 30; i++) {

const day = new Date(thirtyDaysAgo);

day.setDate(thirtyDaysAgo.getDate() + i);

let dailyChange = 0;

const dayString = day.toDateString();

relevantTransactions.forEach(t => {

if (t.date.toDateString() === dayString) {

dailyChange += (t.type === 'income') ? t.amount : -t.amount;

}

});

runningBalance += dailyChange;

// Не позволяем графику уходить в прошлое до первой транзакции

if (i===0) runningBalance = balance30DaysAgo + dailyChange;

chartData.push({ x: day, y: runningBalance });

}

savingsChart = new Chart(ctx, {

type: 'line',

data: {

datasets: [{

label: `Накопления в ${escapeHtml(mainCurrency)}`,

data: chartData,

borderColor: 'var(--brand-accent)',

backgroundColor: 'rgba(167, 139, 250, 0.1)',

fill: true,

tension: 0.4,

pointRadius: 0,

}]

},

options: {

responsive: true, maintainAspectRatio: false,

scales: {

x: { type: 'time', time: { unit: 'day' }, grid: { color: chartGridColor }, ticks: { color: chartColor } },

y: { grid: { color: chartGridColor }, ticks: { color: chartColor, callback: value => formatCurrency(value, mainCurrency) } }

},

plugins: { legend: { display: false } }

}

});

}

// --- УНИВЕРСАЛЬНЫЕ РЕНДЕРЫ ---

function renderGenericPanel(panelId, title, items, itemRenderer, noItemsMessage, addBtnLabel, addBtnAction) {

const panel = document.getElementById(panelId);

if(!panel) return;

panel.innerHTML = `

<div class="bg-card rounded-xl border border-card">

<div class="p-6 flex justify-between items-center border-b border-card">

<h2 class="text-xl font-bold text-main">${escapeHtml(title)}</h2>

<button class="btn-primary" data-action="${escapeHtml(addBtnAction)}">${escapeHtml(addBtnLabel)}</button>

</div>

<div class="p-6 space-y-3">

${items.length > 0 ? items.map(itemRenderer).join('') : `<p class="text-secondary text-center py-8">${escapeHtml(noItemsMessage)}</p>`}

</div>

</div>`;

}

function renderTransactions() {

const panel = document.getElementById('transactions-panel');

if(!panel) return;

const allAccounts = state.allAccounts;

const recurringItemsHTML = state.recurring.length > 0 ? `

<div class="mb-6">

<h3 class="text-lg font-bold mb-3 text-main">Повторяющиеся операции</h3>

<div class="space-y-3">

${state.recurring.map(r => {

const account = allAccounts.find(a => a.id === r.accountId);

return `

<div class="p-4 bg-space-bg rounded-lg flex justify-between items-center">

<div>

<p class="font-bold text-main">${escapeHtml(r.description)}</p>

<p class="text-sm text-secondary">${r.type === 'income' ? 'Доход' : 'Расход'} / ${formatCurrency(r.amount, account?.currency)} / Каждый месяц ${r.dayOfMonth} числа</p>

</div>

<button data-action="delete" data-type="recurring" data-id="${r.id}" class="text-secondary hover:text-red-500">

<svg xmlns="http://www.w3.org/2000/svg" width="20" height="20" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-linejoin="round"><polyline points="3 6 5 6 21 6"></polyline><path d="M19 6v14a2 2 0 0 1-2 2H7a2 2 0 0 1-2-2V6m3 0V4a2 2 0 0 1 2-2h4a2 2 0 0 1 2 2v2"></path></svg>

</button>

</div>`;

}).join('')}

</div>

</div>` : '';

panel.innerHTML = `

<div class="bg-card rounded-xl border border-card">

<div class="p-6">

${recurringItemsHTML}

<div class="flex flex-col sm:flex-row gap-4 justify-between items-center ${state.recurring.length > 0 ? 'border-t border-card pt-6 mt-6' : ''}">

<h2 class="text-xl font-bold text-main">История транзакций</h2>

<div class="flex gap-2">

<button class="btn-secondary" data-action="open-transfer-modal">Перевод</button>

<button class="btn-primary" data-action="open-transaction-modal">Добавить операцию</button>

</div>

</div>

<div class="overflow-x-auto mt-4">

<table class="w-full text-left">

<thead class="text-sm text-secondary">

<tr>

<th class="p-4 font-medium">Дата</th><th class="p-4 font-medium">Категория</th><th class="p-4 font-medium hidden md:table-cell">Счет</th><th class="p-4 font-medium hidden sm:table-cell">Описание</th><th class="p-4 font-medium text-right">Сумма</th><th class="p-4"></th>

</tr>

</thead>

<tbody>

${state.transactions.map(t => {

const account = allAccounts.find(a => a.id === t.accountId);

return `

<tr class="border-t border-card">

<td class="p-4 whitespace-nowrap text-main">${formatDate(t.date)}</td>

<td class="p-4 whitespace-nowrap text-main">${escapeHtml(t.category)}</td>

<td class="p-4 whitespace-nowrap hidden md:table-cell text-main">${account ? escapeHtml(account.name) : 'N/A'}</td>

<td class="p-4 hidden sm:table-cell max-w-xs truncate text-main" title="${escapeHtml(t.description)}">${escapeHtml(t.description) || '-'}</td>

<td class="p-4 font-semibold text-right whitespace-nowrap ${t.type === 'income' ? 'text-green-400' : 'text-red-400'}">

${t.type === 'income' ? '+' : '-'}${formatCurrency(t.amount, account?.currency)}

</td>

<td class="p-4">

<button data-action="delete" data-type="transaction" data-id="${t.id}" class="text-secondary hover:text-red-500">

<svg xmlns="http://www.w3.org/2000/svg" width="20" height="20" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-linejoin="round"><polyline points="3 6 5 6 21 6"></polyline><path d="M19 6v14a2 2 0 0 1-2 2H7a2 2 0 0 1-2-2V6m3 0V4a2 2 0 0 1 2-2h4a2 2 0 0 1 2 2v2"></path></svg>

</button>

</td>

</tr>`;

}).join('') || '<tr><td colspan="6" class="p-6 text-center text-secondary">Транзакций пока нет.</td></tr>'}

</tbody>

</table>

</div>

</div>

</div>`;

}

function renderAccounts() {

const panel = document.getElementById('accounts-panel');

if(!panel) return;

panel.innerHTML = `

<div class="bg-card rounded-xl border border-card">

<div class="p-6 flex justify-between items-center border-b border-card">

<h2 class="text-xl font-bold text-main">Счета и кошельки</h2>

<button class="btn-primary" data-action="open-account-modal">Добавить</button>

</div>

<div class="p-6 space-y-3">

${state.accounts.length > 0 ? state.accounts.map(a => {

const typeLabel = a.type === 'crypto' ? 'Крипто' : 'Счет';

return `<div class="p-4 bg-space-bg rounded-lg flex justify-between items-center">

<div>

<p class="font-bold text-main">${escapeHtml(a.name)} <span class="text-xs font-medium bg-gray-700 text-gray-300 px-2 py-0.5 rounded-full">${typeLabel}</span></p>

<p class="text-sm text-secondary">${formatCurrency(a.balance, a.currency)}</p>

</div>

<button data-action="delete" data-type="account" data-id="${a.id}" class="text-secondary hover:text-red-500"><svg xmlns="http://www.w3.org/2000/svg" width="20" height="20" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-linejoin="round"><polyline points="3 6 5 6 21 6"></polyline><path d="M19 6v14a2 2 0 0 1-2 2H7a2 2 0 0 1-2-2V6m3 0V4a2 2 0 0 1 2-2h4a2 2 0 0 1 2 2v2"></path></svg></button>

</div>`

}).join('') : `<p class="text-secondary text-center py-8">Нет счетов. Добавьте первый.</p>`}

</div>

</div>`;

}

function renderSavings() {

renderGenericPanel('savings-panel', 'Накопительные счета', state.savings,

a => `<div class="p-4 bg-space-bg rounded-lg flex justify-between items-center">

<div><p class="font-bold text-main">${escapeHtml(a.name)}</p><p class="text-sm text-secondary">${formatCurrency(a.balance, a.currency)}</p></div>

<button data-action="delete" data-type="account" data-id="${a.id}" class="text-secondary hover:text-red-500"><svg xmlns="http://www.w3.org/2000/svg" width="20" height="20" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-linejoin="round"><polyline points="3 6 5 6 21 6"></polyline><path d="M19 6v14a2 2 0 0 1-2 2H7a2 2 0 0 1-2-2V6m3 0V4a2 2 0 0 1 2-2h4a2 2 0 0 1 2 2v2"></path></svg></button>

</div>`,

"Нет накопительных счетов.", "Добавить счет", "open-account-modal");

}

function renderPlanner() {

if(!state.currentSpace) return;

const isBusiness = state.currentSpace.type === 'business';

const title = isBusiness ? 'Календарь и Налоги' : 'Календарь платежей';

const items = [...state.plannedPayments, ...(isBusiness ? state.taxes.filter(t => !t.isPaid) : [])]

.sort((a,b) => (a.date || a.dueDate).toMillis() - (b.date || a.dueDate).toMillis());

const itemRenderer = (p) => {

if (p.dueDate) { // Это налог

return `<div class="p-4 bg-space-bg rounded-lg flex justify-between items-center">

<div><p class="font-bold text-main">${escapeHtml(p.name)} - ${formatCurrency(p.amount)}</p><p class="text-sm text-yellow-400">Налог, срок до: ${formatDate(p.dueDate)}</p></div>

<div class="flex gap-2"><button data-action="execute-tax" data-id="${p.id}" class="btn-primary text-xs px-2 py-1">Оплатить</button><button data-action="delete" data-type="tax" data-id="${p.id}" class="text-secondary hover:text-red-500"><svg xmlns="http://www.w3.org/2000/svg" width="20" height="20" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-linejoin="round"><polyline points="3 6 5 6 21 6"></polyline><path d="M19 6v14a2 2 0 0 1-2 2H7a2 2 0 0 1-2-2V6m3 0V4a2 2 0 0 1 2-2h4a2 2 0 0 1 2 2v2"></path></svg></button></div>

</div>`;

} else { // Это запланированный платеж

return `<div class="p-4 bg-space-bg rounded-lg flex justify-between items-center">

<div><p class="font-bold text-main">${escapeHtml(p.description)} - ${formatCurrency(p.amount)}</p><p class="text-sm text-secondary">${formatDate(p.date)} (${escapeHtml(p.category)})</p></div>

<div class="flex gap-2"><button data-action="execute-planned" data-id="${p.id}" class="btn-primary text-xs px-2 py-1">Выполнить</button><button data-action="delete" data-type="plannedPayment" data-id="${p.id}" class="text-secondary hover:text-red-500"><svg xmlns="http://www.w3.org/2000/svg" width="20" height="20" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-linejoin="round"><polyline points="3 6 5 6 21 6"></polyline><path d="M19 6v14a2 2 0 0 1-2 2H7a2 2 0 0 1-2-2V6m3 0V4a2 2 0 0 1 2-2h4a2 2 0 0 1 2 2v2"></path></svg></button></div>

</div>`;

}

};

renderGenericPanel('planner-panel', title, items, itemRenderer, "Нет запланированных операций.", "Запланировать", "open-planned-payment-modal");

}

function renderBudgetsAndGoals() {

const panel = document.getElementById('budgets-goals-panel');

if(!panel) return;

const startOfMonth = new Date(new Date().getFullYear(), new Date().getMonth(), 1);

const budgetsHTML = `

<div class="bg-card rounded-xl border border-card">

<div class="p-6 flex justify-between items-center border-b border-card">

<h2 class="text-xl font-bold text-main">Бюджеты</h2>

<button class="btn-primary" data-action="open-budget-modal">Создать бюджет</button>

</div>

<div class="p-6 space-y-3">

${state.budgets.length > 0 ? state.budgets.map(b => {

const spent = state.transactions

.filter(t => t.type === 'expense' && t.category === b.category && t.date && t.date.toDate() >= startOfMonth)

.reduce((sum, t) => sum + t.amount, 0);

const percentage = b.amount > 0 ? Math.min((spent / b.amount) \* 100, 100) : 0;

return `

<div class="p-4 bg-space-bg rounded-lg">

<div class="flex justify-between items-center mb-2">

<p class="font-bold text-main">${escapeHtml(b.category)}</p>

<button data-action="delete" data-type="budget" data-id="${b.id}" class="text-secondary hover:text-red-500"><svg xmlns="http://www.w3.org/2000/svg" width="20" height="20" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-linejoin="round"><polyline points="3 6 5 6 21 6"></polyline><path d="M19 6v14a2 2 0 0 1-2 2H7a2 2 0 0 1-2-2V6m3 0V4a2 2 0 0 1 2-2h4a2 2 0 0 1 2 2v2"></path></svg></button>

</div>

<div class="flex justify-between text-sm mb-1 text-secondary">

<span>Потрачено: ${formatCurrency(spent)}</span>

<span>Осталось: ${formatCurrency(b.amount - spent)}</span>

</div>

<div class="w-full progress-bar-bg rounded-full h-2">

<div class="progress-bar-fill h-2 rounded-full" style="width: ${percentage}%"></div>

</div>

</div>`;

}).join('') : `<p class="text-secondary text-center py-8">Бюджеты не установлены.</p>`}

</div>

</div>`;

const goalsHTML = `

<div class="bg-card rounded-xl border border-card">

<div class="p-6 flex justify-between items-center border-b border-card">

<h2 class="text-xl font-bold text-main">Цели</h2>

<button class="btn-primary" data-action="open-goal-modal">Добавить цель</button>

</div>

<div class="p-6 space-y-3">

${state.goals.length > 0 ? state.goals.map(g => {

const percentage = g.targetAmount > 0 ? Math.min((g.currentAmount / g.targetAmount) \* 100, 100) : 0;

return `

<div class="p-4 bg-space-bg rounded-lg">

<div class="flex justify-between items-start mb-2">

<div>

<p class="font-bold text-main">${escapeHtml(g.name)}</p>

<p class="text-sm text-secondary">${formatCurrency(g.currentAmount)} / ${formatCurrency(g.targetAmount)}</p>

</div>

<div class="flex items-center gap-2">

<button data-action="open-add-funds-modal" data-id="${g.id}" class="btn-primary text-xs px-2 py-1">Пополнить</button>

<button data-action="delete" data-type="goal" data-id="${g.id}" class="text-secondary hover:text-red-500"><svg xmlns="http://www.w3.org/2000/svg" width="20" height="20" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-linejoin="round"><polyline points="3 6 5 6 21 6"></polyline><path d="M19 6v14a2 2 0 0 1-2 2H7a2 2 0 0 1-2-2V6m3 0V4a2 2 0 0 1 2-2h4a2 2 0 0 1 2 2v2"></path></svg></button>

</div>

</div>

<div class="w-full progress-bar-bg rounded-full h-2">

<div class="progress-bar-fill h-2 rounded-full" style="width: ${percentage}%"></div>

</div>

</div>`;

}).join('') : `<p class="text-secondary text-center py-8">Цели не установлены. Пора мечтать!</p>`}

</div>

</div>`;

panel.innerHTML = `

<div class="grid grid-cols-1 lg:grid-cols-2 gap-6">

<div>${budgetsHTML}</div>

<div>${goalsHTML}</div>

</div>`;

}

function renderSettings() {

const panel = document.getElementById('settings-panel');

if(!panel || !state.currentSpace) return;

const { type, mainCurrency = 'RUB', exchangeRates = {}, categories = defaultCategories, name, taxRate } = state.currentSpace;

const isBusiness = type === 'business';

const otherCurrencies = currencies.filter(c => c !== mainCurrency);

const categoryManagerHTML = (type, title) => `

<div>

<h4 class="text-lg font-bold mb-3 text-main">${escapeHtml(title)}</h4>

<div class="space-y-2">

${categories[type].map(cat => `

<div class="flex justify-between items-center p-2 bg-space-bg rounded">

<span class="text-main">${escapeHtml(cat)}</span>

<button type="button" data-action="delete-category" data-type="${type}" data-category="${escapeHtml(cat)}" class="text-secondary hover:text-red-500 text-2xl leading-none">&times;</button>

</div>

`).join('')}

</div>

<div class="flex gap-2 mt-3">

<input type="text" id="new-category-${type}" placeholder="Новая категория" class="flex-grow">

<button type="button" data-action="add-category" data-type="${type}" class="btn-secondary px-3">Добавить</button>

</div>

</div>`;

panel.innerHTML = `

<div class="bg-card rounded-xl border border-card">

<form id="settings-form">

<div class="p-6 border-b border-card">

<h2 class="text-xl font-bold text-main">Настройки пространства "${escapeHtml(name)}"</h2>

</div>

<div class="p-6 space-y-8">

<div>

<h3 class="text-lg font-bold mb-4 text-main">Категории</h3>

<div class="grid grid-cols-1 md:grid-cols-2 gap-6">

${categoryManagerHTML('income', 'Доходы')}

${categoryManagerHTML('expense', 'Расходы')}

</div>

</div>

<div>

<h3 class="text-lg font-bold mb-3 text-main">Валюта</h3>

<label class="text-sm font-medium text-secondary block mb-1">Основная валюта</label>

<select name="mainCurrency">${currencies.map(c => `<option value="${c}" ${c === mainCurrency ? 'selected' : ''}>${c}</option>`).join('')}</select>

</div>

<div>

<h3 class="text-lg font-bold mb-3 text-main">Курсы обмена</h3>

<p class="text-sm text-secondary mb-4">Укажите, сколько <span class="font-bold text-main">${escapeHtml(mainCurrency)}</span> стоит 1 единица другой валюты.</p>

<div class="space-y-3">

${otherCurrencies.map(c => `

<div class="grid grid-cols-3 items-center gap-4">

<label class="text-right font-medium text-main">1 ${escapeHtml(c)} =</label>

<input type="number" name="rate\_${c}" value="${exchangeRates[c] || ''}" placeholder="0.00" step="0.01" class="col-span-2">

</div>

`).join('')}

</div>

</div>

${isBusiness ? `

<div>

<h3 class="text-lg font-bold mb-3 text-main">Бизнес</h3>

<label class="text-sm font-medium text-secondary block mb-1">Ставка налога, %</label>

<input type="number" name="taxRate" value="${taxRate || 6}" step="0.1" placeholder="6">

</div>` : ''}

<div class="border-t border-card pt-6">

<h3 class="text-lg font-bold text-red-500 mb-3">Опасная зона</h3>

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

<div class="flex justify-between items-center p-4 rounded-lg bg-red-500/10">

<div><p class="font-bold text-main">Экспорт данных</p><p class="text-sm text-secondary">Скачать все данные этого пространства в формате JSON.</p></div>

<button type="button" data-action="export-data" class="btn-secondary">Экспорт</button>

</div>

<div class="flex justify-between items-center p-4 rounded-lg bg-red-500/10">

<div><p class="font-bold text-main">Удалить данные</p><p class="text-sm text-secondary">Удалить все транзакции, счета и т.д. в этом пространстве. Действие необратимо.</p></div>

<button type="button" data-action="delete-space-data" class="bg-red-500 hover:bg-red-600 text-white font-bold py-2 px-4 rounded">Удалить</button>

</div>

</div>

</div>

</div>

<div class="p-6 border-t border-card flex justify-end">

<button type="submit" class="btn-primary">Сохранить изменения</button>

</div>

</form>

</div>`;

}

function renderSpaceSwitcher() {

const container = document.getElementById('space-switcher');

if (!container) return;

const spacesHTML = state.spaces.map(s => {

const isActive = s.id === state.currentSpaceId;

return `<button data-space-id="${s.id}" class="px-3 py-1 text-sm font-medium rounded-md ${isActive ? 'bg-brand-accent text-white' : 'text-space-text-secondary'}">${escapeHtml(s.name)}</button>`;

}).join('');

container.innerHTML = spacesHTML + '<button data-action="open-space-modal" class="px-2 py-1 text-space-text-secondary hover:text-main rounded-md text-xl font-bold">+</button>';

}

// --- УПРАВЛЕНИЕ UI (ТАБЫ, ПРОСТРАНСТВА) ---

function switchTab(tabId) {

document.querySelectorAll('#tab-content > div').forEach(panel => panel.classList.add('hidden'));

document.querySelectorAll('#main-nav button').forEach(tab => tab.classList.remove('tab-active'));

document.getElementById(`${tabId}-panel`)?.classList.remove('hidden');

document.querySelector(`button[data-tab="${tabId}"]`)?.classList.add('tab-active');

}

async function switchSpace(spaceId, isInitialLoad = false) {

if (!isInitialLoad && spaceId === state.currentSpaceId) return;

state.currentSpaceId = spaceId;

state.currentSpace = null;

try {

if (!isInitialLoad) {

await setDoc(doc(db, `artifacts/${appId}/users/${userId}`), { lastUsedSpaceId: spaceId }, { merge: true });

}

renderSpaceSwitcher();

document.getElementById('initial-loader').classList.remove('hidden');

document.getElementById('main-nav').classList.add('hidden');

['dashboard', 'transactions', 'accounts', 'savings', 'planner', 'budgets-goals', 'settings'].forEach(p => {

const el = document.getElementById(`${p}-panel`);

if(el) el.innerHTML = '';

});

await loadDataForCurrentSpace();

} catch (error) {

console.error("Error switching space:", error);

showNotification("Не удалось переключить пространство.", "error");

}

}

// --- ОБРАБОТЧИКИ СОБЫТИЙ ---

function setupEventListeners() {

document.body.addEventListener('click', (e) => {

const target = e.target;

// [UX] Закрытие модального окна по клику на фон

if (target.matches('.fixed[data-modal-id]')) {

hideModal(target.dataset.modalId);

return;

}

const actionTarget = target.closest('[data-action]');

const tabTarget = target.closest('[data-tab]');

const spaceTarget = target.closest('[data-space-id]');

if (actionTarget) handleActionClick(actionTarget);

else if (tabTarget) switchTab(tabTarget.dataset.tab);

else if (spaceTarget) switchSpace(spaceTarget.dataset.spaceId);

});

// [UX] Закрытие модального окна по Escape

document.addEventListener('keydown', (e) => {

if (e.key === "Escape") {

document.querySelectorAll('.fixed[data-modal-id]:not(.hidden)').forEach(modal => {

hideModal(modal.id);

});

}

});

document.body.addEventListener('submit', handleFormSubmit);

}

function handleActionClick(target) {

const action = target.dataset.action;

const id = target.dataset.id;

const type = target.dataset.type;

switch (action) {

case 'open-transaction-modal': {

const form = document.getElementById('transaction-form');

form.reset();

form.querySelector('[name="date"]').value = getISODate();

form.querySelector('[name="accountId"]').innerHTML = allAccountsOptions();

const typeSelect = form.querySelector('[name="type"]');

const categorySelect = form.querySelector('[name="category"]');

const updateCats = () => { categorySelect.innerHTML = typeSelect.value === 'income' ? incomeCategoriesOptions() : expenseCategoriesOptions(); };

// [FIX] Переподключаем слушатель, чтобы избежать дублей

typeSelect.onchange = updateCats;

updateCats();

showModal('transaction-modal');

break;

}

case 'open-transfer-modal': {

const form = document.getElementById('transfer-form');

form.reset();

form.querySelector('[name="date"]').value = getISODate();

form.querySelector('[name="fromAccountId"]').innerHTML = allAccountsOptions();

form.querySelector('[name="toAccountId"]').innerHTML = allAccountsOptions();

showModal('transfer-modal');

break;

}

case 'open-account-modal': {

const form = document.getElementById('account-form');

form.reset();

form.querySelector('[name="currency"]').innerHTML = allCurrenciesOptions();

showModal('account-modal');

break;

}

case 'open-budget-modal': {

const form = document.getElementById('budget-form');

form.reset();

form.querySelector('[name="category"]').innerHTML = expenseCategoriesOptions();

showModal('budget-modal');

break;

}

case 'open-planned-payment-modal': {

const form = document.getElementById('planned-payment-form');

form.reset();

form.querySelector('[name="accountId"]').innerHTML = allAccountsOptions();

form.querySelector('[name="category"]').innerHTML = expenseCategoriesOptions();

showModal('planned-payment-modal');

break;

}

case 'open-space-modal': {

const form = document.getElementById('space-form');

form.reset();

form.querySelector('[name="mainCurrency"]').innerHTML = allCurrenciesOptions();

showModal('space-modal');

break;

}

case 'open-add-funds-modal': {

const form = document.getElementById('add-funds-form');

form.reset();

document.getElementById('add-funds-goal-id').value = id;

form.querySelector('[name="accountId"]').innerHTML = allAccountsOptions();

showModal('add-funds-modal');

break;

}

case 'open-goal-modal':

document.getElementById('goal-form').reset();

showModal('goal-modal');

break;

case 'close-modal':

target.closest('.fixed').classList.add('hidden');

break;

case 'delete':

showConfirmModal(`Вы уверены, что хотите удалить этот элемент?`, () => deleteItem(id, type));

break;

case 'delete-category':

handleCategoryChange('delete', target.dataset.type, target.dataset.category);

break;

case 'add-category': {

const input = document.getElementById(`new-category-${target.dataset.type}`);

if(input.value.trim()) {

handleCategoryChange('add', target.dataset.type, input.value.trim());

input.value = '';

}

break;

}

case 'export-data': exportData(); break;

case 'delete-space-data':

showConfirmModal(`Вы уверены, что хотите удалить ВСЕ ДАННЫЕ в этом пространстве? Это действие необратимо!`, () => deleteSpaceData());

break;

case 'execute-planned': executePlannedPayment(id); break;

case 'execute-tax': executeTaxPayment(id); break;

}

}

async function handleFormSubmit(e) {

e.preventDefault();

const form = e.target;

const submitButton = form.querySelector('button[type="submit"]');

if (!submitButton) return;

const originalText = submitButton.innerHTML;

submitButton.disabled = true;

submitButton.innerHTML = 'Сохранение...';

try {

switch(form.id) {

case 'transaction-form': await saveTransaction(form); break;

case 'transfer-form': await saveTransfer(form); break;

case 'account-form': await saveGeneric(form, 'account', 'accountId'); break;

case 'goal-form': await saveGeneric(form, 'goal', 'goalId'); break;

case 'add-funds-form': await addFundsToGoal(form); break;

case 'planned-payment-form': await saveGeneric(form, 'plannedPayment', 'plannedPaymentId'); break;

case 'budget-form': await saveGeneric(form, 'budget', 'budgetId'); break;

case 'settings-form': await saveSettings(form); break;

case 'space-form': await saveSpace(form); break;

}

} finally {

submitButton.disabled = false;

submitButton.innerHTML = originalText;

}

}

// --- ЛОГИКА СОХРАНЕНИЯ И УДАЛЕНИЯ ---

function showConfirmModal(message, onConfirm) {

const modal = document.getElementById('confirm-modal');

modal.querySelector('#confirm-message').textContent = message;

showModal('confirm-modal');

const confirmBtn = modal.querySelector('#confirm-btn');

// Клонируем кнопку, чтобы удалить старые слушатели

const newConfirmBtn = confirmBtn.cloneNode(true);

confirmBtn.parentNode.replaceChild(newConfirmBtn, confirmBtn);

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

onConfirm();

hideModal('confirm-modal');

}, { once: true });

}

async function deleteItem(id, type) {

if (!userId || !state.currentSpaceId) return;

try {

if (type === 'transaction') {

await runTransaction(db, async (transaction) => {

const txDocRef = doc(db, `artifacts/${appId}/users/${userId}/transactions/${id}`);

const txDocSnap = await transaction.get(txDocRef);

if (!txDocSnap.exists()) throw new Error("Транзакция не найдена");

const tx = txDocSnap.data();

const accountRef = doc(db, `artifacts/${appId}/users/${userId}/accounts/${tx.accountId}`);

const adjustment = tx.type === 'income' ? -tx.amount : tx.amount;

transaction.update(accountRef, { balance: increment(adjustment) });

transaction.delete(txDocRef);

});

} else if (type === 'account') {

const q = query(collection(db, `artifacts/${appId}/users/${userId}/transactions`), where("accountId", "==", id), where("spaceId", "==", state.currentSpaceId));

const transactionsForAccount = await getDocs(q);

if (!transactionsForAccount.empty) {

showNotification('Нельзя удалить счет, к которому привязаны транзакции.', 'error');

return;

}

await deleteDoc(doc(db, `artifacts/${appId}/users/${userId}/accounts/${id}`));

} else {

await deleteDoc(doc(db, `artifacts/${appId}/users/${userId}/${type}s/${id}`));

}

showNotification('Элемент удален', 'success');

} catch (error) {

console.error(`Error deleting ${type}:`, error);

showNotification(`Ошибка удаления: ${error.message}`, 'error');

}

}

async function saveTransaction(form) {

const formData = new FormData(form);

const type = formData.get('type');

const amount = parseFloat(formData.get('amount'));

const accountId = formData.get('accountId');

if (!amount || amount <= 0 || !accountId) {

showNotification('Сумма должна быть положительной, и счет должен быть выбран.', 'error');

return;

}

try {

await runTransaction(db, async (transaction) => {

const accountRef = doc(db, `artifacts/${appId}/users/${userId}/accounts`, accountId);

const txRef = doc(collection(db, `artifacts/${appId}/users/${userId}/transactions`));

const newTransaction = {

type, amount, accountId,

category: formData.get('category'),

description: formData.get('description'),

date: Timestamp.fromDate(new Date(formData.get('date'))),

spaceId: state.currentSpaceId,

createdAt: serverTimestamp()

};

const adjustment = type === 'income' ? amount : -amount;

transaction.update(accountRef, { balance: increment(adjustment) });

transaction.set(txRef, newTransaction);

if (state.currentSpace.type === 'business' && type === 'income') {

const taxRate = state.currentSpace.taxRate || 0;

const taxAmount = amount \* (taxRate / 100);

const spaceRef = doc(db, `artifacts/${appId}/users/${userId}/spaces`, state.currentSpaceId);

transaction.update(spaceRef, { taxProvision: increment(taxAmount) });

}

});

showNotification('Транзакция сохранена', 'success');

hideModal('transaction-modal');

} catch (error) {

console.error("Transaction failed:", error);

showNotification(`Ошибка сохранения: ${error.message}`, 'error');

}

}

async function saveTransfer(form) {

const formData = new FormData(form);

const amount = parseFloat(formData.get('amount'));

const fromAccountId = formData.get('fromAccountId');

const toAccountId = formData.get('toAccountId');

const date = Timestamp.fromDate(new Date(formData.get('date')));

const description = formData.get('description') || 'Перевод между счетами';

if (!amount || amount <= 0 || !fromAccountId || !toAccountId) {

showNotification('Сумма должна быть положительной, и оба счета должны быть выбраны.', 'error');

return;

}

if(fromAccountId === toAccountId) {

showNotification('Нельзя перевести деньги на тот же счет.', 'error');

return;

}

const fromAccount = state.allAccounts.find(a => a.id === fromAccountId);

const toAccount = state.allAccounts.find(a => a.id === toAccountId);

if (fromAccount && toAccount && fromAccount.currency !== toAccount.currency) {

showNotification(`Переводы возможны только между счетами в одной валюте.`, 'error');

return;

}

try {

await runTransaction(db, async (transaction) => {

const fromAccountRef = doc(db, `artifacts/${appId}/users/${userId}/accounts`, fromAccountId);

const toAccountRef = doc(db, `artifacts/${appId}/users/${userId}/accounts`, toAccountId);

transaction.update(fromAccountRef, { balance: increment(-amount) });

transaction.update(toAccountRef, { balance: increment(amount) });

const expenseTxRef = doc(collection(db, `artifacts/${appId}/users/${userId}/transactions`));

const incomeTxRef = doc(collection(db, `artifacts/${appId}/users/${userId}/transactions`));

const commonData = { amount, description, date, spaceId: state.currentSpaceId, createdAt: serverTimestamp(), category: 'Перевод' };

transaction.set(expenseTxRef, { ...commonData, type: 'expense', accountId: fromAccountId });

transaction.set(incomeTxRef, { ...commonData, type: 'income', accountId: toAccountId });

});

showNotification('Перевод выполнен', 'success');

hideModal('transfer-modal');

} catch(error) {

console.error("Transfer failed:", error);

showNotification(`Ошибка перевода: ${error.message}`, 'error');

}

}

async function saveGeneric(form, collectionName, idField) {

const formData = new FormData(form);

const id = formData.get(idField);

let data = Object.fromEntries(formData.entries());

// [STABILITY] Надежное преобразование типов

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

const numericFields = ['amount', 'balance', 'targetAmount', 'currentAmount'];

if (numericFields.includes(key) && typeof data[key] === 'string' && data[key].trim() !== '') {

data[key] = parseFloat(data[key]);

}

});

if(data.date) data.date = Timestamp.fromDate(new Date(data.date));

delete data[idField];

data.spaceId = state.currentSpaceId;

try {

const collectionRef = collection(db, `artifacts/${appId}/users/${userId}/${collectionName}s`);

if (id) {

await setDoc(doc(collectionRef, id), data, { merge: true });

} else {

data.createdAt = serverTimestamp();

await addDoc(collectionRef, data);

}

showNotification('Данные сохранены', 'success');

hideModal(`${collectionName}-modal`);

} catch (error) {

console.error(`Error saving ${collectionName}:`, error);

showNotification(`Ошибка сохранения: ${error.message}`, 'error');

}

}

async function saveSpace(form) {

const formData = new FormData(form);

const name = formData.get('name').trim();

if(!name) {

showNotification('Название пространства не может быть пустым.', 'error');

return;

}

const data = {

name: name,

type: formData.get('type'),

mainCurrency: formData.get('mainCurrency'),

createdAt: serverTimestamp(),

categories: defaultCategories,

exchangeRates: {},

};

try {

const newSpaceRef = await addDoc(collection(db, `artifacts/${appId}/users/${userId}/spaces`), data);

await switchSpace(newSpaceRef.id);

showNotification('Пространство создано', 'success');

hideModal('space-modal');

} catch(e) {

console.error("Error saving space:", e);

showNotification('Ошибка сохранения пространства.', 'error');

}

}

async function addFundsToGoal(form) {

const formData = new FormData(form);

const amount = parseFloat(formData.get('amount'));

const accountId = formData.get('accountId');

const goalId = formData.get('goalId');

if (!amount || amount <= 0 || !accountId || !goalId) {

showNotification('Сумма должна быть положительной, и счет должен быть выбран.', 'error');

return;

}

try {

await runTransaction(db, async (transaction) => {

const accountRef = doc(db, `artifacts/${appId}/users/${userId}/accounts`, accountId);

const goalRef = doc(db, `artifacts/${appId}/users/${userId}/goals`, goalId);

const txRef = doc(collection(db, `artifacts/${appId}/users/${userId}/transactions`));

transaction.update(accountRef, { balance: increment(-amount) });

transaction.update(goalRef, { currentAmount: increment(amount) });

const goal = state.goals.find(g => g.id === goalId);

transaction.set(txRef, {

type: 'expense', amount, accountId, category: 'Цели',

description: `Пополнение цели: ${goal?.name || ''}`,

date: Timestamp.now(), spaceId: state.currentSpaceId, createdAt: serverTimestamp()

});

});

showNotification('Цель пополнена', 'success');

hideModal('add-funds-modal');

} catch(error) {

console.error("Adding funds failed:", error);

showNotification(`Ошибка пополнения: ${error.message}`, 'error');

}

}

async function saveSettings(form) {

const formData = new FormData(form);

const mainCurrency = formData.get('mainCurrency');

const taxRate = parseFloat(formData.get('taxRate')) || 0;

const exchangeRates = {};

for (const [key, value] of formData.entries()) {

if (key.startsWith('rate\_') && value) {

exchangeRates[key.replace('rate\_', '')] = parseFloat(value);

}

}

try {

const spaceRef = doc(db, `artifacts/${appId}/users/${userId}/spaces`, state.currentSpaceId);

const updateData = { mainCurrency, exchangeRates };

if(state.currentSpace.type === 'business') {

updateData.taxRate = taxRate;

}

await setDoc(spaceRef, updateData, { merge: true });

showNotification('Настройки сохранены', 'success');

} catch(e) {

console.error("Error saving settings:", e);

showNotification('Ошибка сохранения настроек', 'error');

}

}

async function handleCategoryChange(changeType, categoryType, categoryValue) {

if(!state.currentSpace || !categoryValue) return;

const currentCategories = JSON.parse(JSON.stringify(state.currentSpace.categories || defaultCategories));

if(changeType === 'add') {

if (!currentCategories[categoryType].includes(categoryValue)) {

currentCategories[categoryType].push(categoryValue);

}

} else if (changeType === 'delete') {

currentCategories[categoryType] = currentCategories[categoryType].filter(c => c !== categoryValue);

}

try {

const spaceRef = doc(db, `artifacts/${appId}/users/${userId}/spaces`, state.currentSpaceId);

await setDoc(spaceRef, { categories: currentCategories }, { merge: true });

showNotification('Категории обновлены', 'success');

} catch (error) {

console.error("Error updating categories:", error);

showNotification('Ошибка обновления категорий', 'error');

}

}

function exportData() {

if (!state.currentSpace) {

showNotification("Нет данных для экспорта.", "error");

return;

}

const dataToExport = { ...state, unsubscribers: undefined, allAccounts: undefined };

const dataStr = "data:text/json;charset=utf-8," + encodeURIComponent(JSON.stringify(dataToExport, (key, value) => {

if (value && value.toDate) return value.toDate().toISOString(); // Сериализация Timestamps

return value;

}, 2));

const downloadAnchorNode = document.createElement('a');

downloadAnchorNode.setAttribute("href", dataStr);

downloadAnchorNode.setAttribute("download", `finance\_export\_${state.currentSpace.name}\_${new Date().toISOString().split('T')[0]}.json`);

document.body.appendChild(downloadAnchorNode);

downloadAnchorNode.click();

downloadAnchorNode.remove();

showNotification('Данные экспортированы', 'success');

}

async function deleteSpaceData() {

const collectionsToDelete = ['transactions', 'goals', 'budgets', 'plannedPayments', 'recurring', 'taxes', 'accounts'];

showNotification("Начинаем удаление данных...", "info");

try {

for (const colName of collectionsToDelete) {

const q = query(collection(db, `artifacts/${appId}/users/${userId}/${colName}`), where("spaceId", "==", state.currentSpaceId));

const snapshot = await getDocs(q);

if (snapshot.empty) continue;

const batch = writeBatch(db);

snapshot.docs.forEach(d => batch.delete(d.ref));

await batch.commit();

}

showNotification('Все данные пространства удалены.', 'success');

} catch (error) {

console.error("Error deleting space data:", error);

showNotification('Ошибка при удалении данных.', 'error');

}

}

async function executePlannedPayment(id) {

const payment = state.plannedPayments.find(p => p.id === id);

if (!payment) return;

// Используем фиктивную форму для передачи данных в saveTransaction

const form = document.createElement('form');

form.innerHTML = `

<input name="type" value="expense">

<input name="amount" value="${payment.amount}">

<input name="accountId" value="${payment.accountId}">

<input name="category" value="${payment.category}">

<input name="description" value="${payment.description}">

<input name="date" value="${getISODate(payment.date)}">

`;

await saveTransaction(form);

await deleteItem(id, 'plannedPayment');

}

async function executeTaxPayment(id) {

const tax = state.taxes.find(t => t.id === id);

if (!tax || !state.currentSpace) return;

const mainCurrencyAccounts = state.accounts.filter(a => a.currency === state.currentSpace.mainCurrency);

if(mainCurrencyAccounts.length === 0) {

showNotification(`Нет счетов в основной валюте (${escapeHtml(state.currentSpace.mainCurrency)}) для оплаты налога.`, 'error');

return;

}

const accountId = mainCurrencyAccounts[0].id;

try {

await runTransaction(db, async (transaction) => {

const accountRef = doc(db, `artifacts/${appId}/users/${userId}/accounts`, accountId);

const txRef = doc(collection(db, `artifacts/${appId}/users/${userId}/transactions`));

const taxRef = doc(db, `artifacts/${appId}/users/${userId}/taxes`, id);

const spaceRef = doc(db, `artifacts/${appId}/users/${userId}/spaces`, state.currentSpaceId);

transaction.update(accountRef, { balance: increment(-tax.amount) });

transaction.update(spaceRef, { taxProvision: increment(-tax.amount) });

transaction.update(taxRef, { isPaid: true });

transaction.set(txRef, { type: 'expense', amount: tax.amount, accountId, category: 'Налоги', description: `Оплата налога: ${tax.name}`, date: Timestamp.now(), spaceId: state.currentSpaceId });

});

showNotification('Налог оплачен', 'success');

} catch(e) {

console.error("Tax payment failed:", e);

showNotification(`Ошибка оплаты налога: ${e.message}`, 'error');

}

}

// [REFACTOR] Весь HTML-код для модальных окон инкапсулирован здесь

function injectModalSkeletons() {

const placeholder = document.getElementById('modal-placeholders');

if (!placeholder) return;

placeholder.innerHTML = `

<!-- Универсальное модальное окно подтверждения -->

<div id="confirm-modal" data-modal-id="confirm-modal" class="fixed inset-0 bg-black/70 flex items-center justify-center hidden z-50 p-4">

<div class="bg-card rounded-xl border border-card shadow-glow-lg w-full max-w-sm">

<div class="p-6">

<h3 class="text-lg font-bold text-main mb-4">Подтверждение</h3>

<p id="confirm-message" class="text-secondary mb-6">Вы уверены?</p>

<div class="flex justify-end gap-3">

<button data-action="close-modal" class="btn-secondary">Отмена</button>

<button id="confirm-btn" class="bg-red-500 hover:bg-red-600 text-white font-bold py-2 px-4 rounded">Подтвердить</button>

</div>

</div>

</div>

</div>

<!-- Модальные окна для форм -->

${createModal('space-modal', 'Новое рабочее пространство', `

<form id="space-form">

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

<div><label class="text-sm font-medium text-secondary block mb-1">Название</label><input type="text" name="name" placeholder="Например, Личные финансы" required></div>

<div><label class="text-sm font-medium text-secondary block mb-1">Тип</label><select name="type"><option value="personal">Личное</option><option value="business">Бизнес</option></select></div>

<div><label class="text-sm font-medium text-secondary block mb-1">Основная валюта</label><select name="mainCurrency"></select></div>

</div>

<div class="mt-6 flex justify-end"><button type="submit" class="btn-primary">Создать</button></div>

</form>

`)}

${createModal('transaction-modal', 'Новая транзакция', `

<form id="transaction-form">

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

<div><label class="text-sm font-medium text-secondary block mb-1">Тип</label><select name="type"><option value="expense">Расход</option><option value="income">Доход</option></select></div>

<div><label class="text-sm font-medium text-secondary block mb-1">Сумма</label><input type="number" name="amount" placeholder="0.00" required step="0.01"></div>

<div class="md:col-span-2"><label class="text-sm font-medium text-secondary block mb-1">Счет</label><select name="accountId" required></select></div>

<div><label class="text-sm font-medium text-secondary block mb-1">Категория</label><select name="category" required></select></div>

<div><label class="text-sm font-medium text-secondary block mb-1">Дата</label><input type="date" name="date" required></div>

<div class="md:col-span-2"><label class="text-sm font-medium text-secondary block mb-1">Описание</label><input type="text" name="description" placeholder="Покупка в магазине"></div>

</div>

<div class="mt-6 flex justify-end"><button type="submit" class="btn-primary">Сохранить</button></div>

</form>

`)}

${createModal('transfer-modal', 'Перевод между счетами', `

<form id="transfer-form">

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

<div><label class="text-sm font-medium text-secondary block mb-1">Со счета</label><select name="fromAccountId" required></select></div>

<div><label class="text-sm font-medium text-secondary block mb-1">На счет</label><select name="toAccountId" required></select></div>

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

<div><label class="text-sm font-medium text-secondary block mb-1">Сумма</label><input type="number" name="amount" placeholder="0.00" required step="0.01"></div>

<div><label class="text-sm font-medium text-secondary block mb-1">Дата</label><input type="date" name="date" required></div>

</div>

<div><label class="text-sm font-medium text-secondary block mb-1">Комментарий</label><input type="text" name="description" placeholder="Перевод"></div>

</div>

<div class="mt-6 flex justify-end"><button type="submit" class="btn-primary">Выполнить перевод</button></div>

</form>

`)}

${createModal('account-modal', 'Новый счет', `

<form id="account-form">

<input type="hidden" name="accountId">

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

<div><label class="text-sm font-medium text-secondary block mb-1">Название счета</label><input type="text" name="name" placeholder="Наличные" required></div>

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

<div><label class="text-sm font-medium text-secondary block mb-1">Начальный баланс</label><input type="number" name="balance" value="0" required step="0.01"></div>

<div><label class="text-sm font-medium text-secondary block mb-1">Валюта</label><select name="currency" required></select></div>

</div>

<div><label class="text-sm font-medium text-secondary block mb-1">Тип счета</label><select name="type"><option value="regular">Обычный</option><option value="savings">Накопительный</option><option value="crypto">Крипто</option></select></div>

</div>

<div class="mt-6 flex justify-end"><button type="submit" class="btn-primary">Добавить счет</button></div>

</form>

`)}

${createModal('goal-modal', 'Новая цель', `

<form id="goal-form">

<input type="hidden" name="goalId">

<input type="hidden" name="currentAmount" value="0">

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

<div><label class="text-sm font-medium text-secondary block mb-1">Название цели</label><input type="text" name="name" placeholder="Отпуск" required></div>

<div><label class="text-sm font-medium text-secondary block mb-1">Целевая сумма</label><input type="number" name="targetAmount" placeholder="100000" required></div>

</div>

<div class="mt-6 flex justify-end"><button type="submit" class="btn-primary">Создать цель</button></div>

</form>

`)}

${createModal('add-funds-modal', 'Пополнить цель', `

<form id="add-funds-form">

<input type="hidden" name="goalId" id="add-funds-goal-id">

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

<div><label class="text-sm font-medium text-secondary block mb-1">Сумма пополнения</label><input type="number" name="amount" placeholder="0.00" required step="0.01"></div>

<div><label class="text-sm font-medium text-secondary block mb-1">Списать со счета</label><select name="accountId" required></select></div>

</div>

<div class="mt-6 flex justify-end"><button type="submit" class="btn-primary">Пополнить</button></div>

</form>

`)}

${createModal('planned-payment-modal', 'Запланировать операцию', `

<form id="planned-payment-form">

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

<div><label class="text-sm font-medium text-secondary block mb-1">Описание</label><input type="text" name="description" placeholder="Аренда" required></div>

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

<div><label class="text-sm font-medium text-secondary block mb-1">Сумма</label><input type="number" name="amount" placeholder="0.00" required step="0.01"></div>

<div><label class="text-sm font-medium text-secondary block mb-1">Дата</label><input type="date" name="date" required></div>

</div>

<div><label class="text-sm font-medium text-secondary block mb-1">Категория</label><select name="category" required></select></div>

<div><label class="text-sm font-medium text-secondary block mb-1">Счет для списания</label><select name="accountId" required></select></div>

</div>

<div class="mt-6 flex justify-end"><button type="submit" class="btn-primary">Запланировать</button></div>

</form>

`)}

${createModal('budget-modal', 'Новый бюджет', `

<form id="budget-form">

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

<div><label class="text-sm font-medium text-secondary block mb-1">Категория расходов</label><select name="category" required></select></div>

<div><label class="text-sm font-medium text-secondary block mb-1">Месячный лимит</label><input type="number" name="amount" placeholder="0.00" required step="0.01"></div>

</div>

<div class="mt-6 flex justify-end"><button type="submit" class="btn-primary">Сохранить бюджет</button></div>

</form>

`)}

`;

}

// Помощник для создания разметки модального окна

function createModal(id, title, formHtml) {

return `

<div id="${id}" data-modal-id="${id}" class="fixed inset-0 bg-black/70 flex items-center justify-center hidden z-50 p-4">

<div class="bg-card rounded-xl border border-card shadow-glow-lg w-full max-w-lg modal-container">

<div class="p-6">

<div class="flex justify-between items-center mb-6">

<h3 class="text-xl font-bold text-main">${title}</h3>

<button data-action="close-modal" class="text-secondary hover:text-main">&times;</button>

</div>

${formHtml}

</div>

</div>

</div>`;

}

</script>

</body>

</html>