Project Dynamic Data Switching - Implementation Summary

Finanzas SD – Architecture, Flows & SOPs

Arquitectura, Flujos y Procedimientos

November 11, 2025

1 Project Dynamic Data Switching - Implementation Summary

1.1 Enhanced Project Context Management

1.1.1 1. ProjectContext Improvements

- ✅ Added `projectChangeCount` tracking for debugging
- ✅ Enhanced project setter with logging
- ✅ Added useEffect to log project changes

1.1.2 2. Component Updates - All SDMT modules now properly respond to project changes

Forecast Component (`SDMTForecast.tsx`)

- ✅ Connected to `useProject()` context
- ✅ Loads data on project/period change via `useEffect`
- ✅ Shows project name in header
- ✅ Added loading states with project info
- ✅ Console logging for debugging data refreshes

Catalog Component (`SDMTCatalog.tsx`)

- ✅ Connected to `useProject()` context
- ✅ Loads line items when project changes
- ✅ Console logging for debugging

Reconciliation Component (`SDMTReconciliation.tsx`)

- ✅ Connected to `useProject()` context

- ✅ Loads invoices and line items on project change
- ✅ Console logging for debugging

Cashflow Component (`SDMTCashflow.tsx`)

- ✅ Connected to `useProject()` context
- ✅ Loads cashflow data from API per project
- ✅ Fallback to mock data if API fails
- ✅ Console logging for debugging

Scenarios Component (`SDMTScenarios.tsx`)

- ✅ Connected to `useProject()` context
- ✅ Loads scenarios per project via API
- ✅ Loading state with project info
- ✅ Console logging for debugging

Changes Component (`SDMTChanges.tsx`)

- ✅ Connected to `useProject()` context
- ✅ Loads change requests per project
- ✅ Proper type mapping from domain types
- ✅ Shows project name in header
- ✅ Console logging for debugging

1.1.3 3. API Service Enhancements

The API service already had proper project-specific data handling: - ✅ `getLineItems(project_id)` - Returns different line items per project - ✅ `getForecastData(project_id, months)` - Returns project-specific forecast data - ✅ `getInvoices(project_id)` - Returns project-specific invoices - ✅ `getCashFlowData(project_id, months)` - Calculates per-project cashflows - ✅ `getScenarios(project_id)` - Returns project scenarios - ✅ `getChangeRequests(project_id)` - Returns project change requests

1.1.4 4. Mock Data Structure

Each project has its own dedicated mock data files: - `baseline.json` - Healthcare System Modernization project - `baseline-fintech.json` - Banking Core Platform project - `baseline-retail.json` - Retail Intelligence & Analytics project - Similar structure for forecast, invoices, and billing-plan data

1.1.5 5. User Experience Improvements

- ✅ Loading states show which project data is being loaded
- ✅ Project name displayed in component headers
- ✅ Console logging helps debug data loading issues
- ✅ ProjectContextBar allows easy project switching
- ✅ All data refreshes automatically when project selection changes

1.2 How It Works

1. User selects a project in the ProjectContextBar dropdown

2. `setSelectedProjectId()` is called, triggering `projectChangeCount` increment
3. All SDMT components have `useEffect` dependencies on `[selectedProjectId, projectChangeCount]`
4. Each component's `loadData()` function is called with the new `selectedProjectId`
5. API service returns project-specific data from appropriate mock files
6. UI updates to show new project's data
7. Console logs provide visibility into the refresh process

1.3 Testing the Implementation

To verify dynamic switching works:

1. Open browser dev tools console
2. Navigate to any SDMT tab (Catalog, Forecast, etc.)
3. Change project selection in the dropdown
4. Watch console logs showing data refresh for each component
5. Verify data in tables/charts changes to reflect the selected project
6. Switch between different projects to confirm each has distinct data

The implementation ensures seamless data switching across all tabs without page refreshes or component re-mounts.