Baseline Data Lineage Overview

Finanzas SD – Architecture, Flows & SOPs

Arquitectura, Flujos y Procedimientos

December 11, 2025

1 Baseline Data Lineage Overview

1.1 Purpose

This document explains the complete data lineage from PMO Estimator through Baseline to SDMT cost management, ensuring proper tracking and materialization of labor and non-labor costs.

1.2 Architecture Overview

```
┌─────────────────┐
│ PMO Estimator   │  User creates project estimates
│  (Frontend)     │  - Labor: MOD roles with rates
│                 │  - Non-labor: Rubros from catalog
└────────┬────────┘
         │ Creates baseline with canonical rubroIds
         v
┌─────────────────┐
│ Baseline API    │  Stores estimates in prefacturas table
│  (Backend)      │  - Labor estimates with rubroId (MOD-xxx)
│                 │  - Non-labor estimates with rubroId (GSV-xxx, etc.)
└────────┬────────┘
         │ Handoff creates project and materializes rubros
         v
┌─────────────────┐
│ Project Rubros  │  Rubros table entries with baseline_id
│  (DynamoDB)     │  - PK: PROJECT#{projectId}
│                 │  - SK: RUBRO#{rubroId}#{baselineId}#{index}
│                 │  - metadata.linea_codigo: canonical taxonomy code
└────────┬────────┘
         │ Filtered by active baseline_id
         v
┌─────────────────┐
│ SDMT Forecast   │  Monthly P/F/A grid for cost management
│  (Frontend)     │  - Baseline-filtered rubros only
│                 │  - Cost allocation by month
│                 │  - Variance tracking
└─────────────────┘
```

1.3 Key Components

1.3.1 1. Canonical Rubros Taxonomy

Location: `src/modules/rubros.taxonomia.ts`, `src/modules/rubros.catalog.ts`
The rubros taxonomy provides the single source of truth for all cost categories:

- **MOD (Mano de Obra Directa)**: Labor categories
  - `MOD-ING`: Support Engineers (N1/N2/N3)
  - `MOD-LEAD`: Delivery Engineers (Lead/Coordinator)
  - `MOD-SDM`: Service Delivery Manager
  - `MOD-PM`: Project Manager
  - etc.
- **GSV (Gestión del Servicio)**: Service management
- **SOI (Soporte e Infraestructura)**: Infrastructure support
- **SEC (Seguridad y Cumplimiento)**: Security and compliance
- And more…

Each rubro has: - **linea_codigo**: Canonical ID (e.g., "MOD-ING") - **linea_gasto**: Display label - **categoria_codigo**: Category code - **tipo_ejecucion**: "mensual" or "puntual/hito" - **tipo_costo**: "OPEX" or "CAPEX"

1.3.2 2. PMO Estimator

Location: `src/features/pmo/prefactura/Estimator/`
The Estimator UI allows users to create project estimates:

Labor Step (`LaborStep.tsx`)

- Users select MOD roles from dropdown (e.g., "Ingeniero Delivery")
- **CRITICAL**: Component stores canonical `rubroId` (e.g., "MOD-LEAD") not display string
- Maps display role → rubroId using `mapModRoleToRubroId()`
- Stores: `{ rubroId: "MOD-LEAD", role: "Ingeniero Delivery", ...rates... }`

Non-Labor Step (`NonLaborStep.tsx`)

- Users select rubros from categorized dropdown (organized by category)
- **CRITICAL**: Component stores canonical `rubroId` (e.g., "GSV-REU") not free-form category
- Auto-populates category and description based on selected rubroId
- Stores: `{ rubroId: "GSV-REU", category: "Gestión del Servicio", description: "..." }`

1.3.3 3. Baseline Creation

Location: `services/finanzas-api/src/handlers/baseline.ts`
When user submits the Estimator:

1. Frontend sends POST to `/baseline` with:

```
[] { ”project_name”: ”Project X”, ”labor_estimates”: [ { ”rubroId”: ”MOD-LEAD”,
”role”: ”Ingeniero Delivery”, ”country”: ”Colombia”, ”level”: ”senior”, ”fte_count”:
1, ”hourly_rate”: 6000, ... } ], ”non_labor_estimates”: [ { ”rubroId”: ”GSV-REU”,
”category”: ”Gestión del Servicio”, ”description”: ”Reuniones de seguimiento”,
”amount”: 1000, ... } ] }
```

2. Backend creates baseline record in `prefacturas` table:
   - PK: `PROJECT#{projectId}`
   - SK: `BASELINE#{baselineId}`
   - Stores full labor_estimates and non_labor_estimates arrays

3. Backend also creates metadata record:
   - PK: `BASELINE#{baselineId}`
   - SK: `METADATA`
   - For lookup and listing

1.3.4 4. Handoff & Rubro Materialization

Location: `services/finanzas-api/src/handlers/handoff.ts`, `services/finanzas-api/src/h`
When baseline is handed off to SDMT:

1. **Handoff creates project metadata** (`POST /projects/{projectId}/handoff`):
   - Updates projects table with baseline_id
   - Sets baseline_status to “handed_off” or “accepted”
   - Links project to baseline for filtering

2. **Rubros are materialized** (`generateLineItems()` in `projects.ts`):

   For each labor estimate:

```
[] { rubroId: "MOD-LEAD#base_abc123#1", // Unique SK nombre: "Ingeniero Delivery", category: "Labor", qty: 1, unit_cost: 7500, // monthly cost with on-costs recurring: true, start_month: 1, end_month: 12, metadata: { source: "baseline", baseline_id: "base_abc123", project_id: "PRJ-X", linea_codigo: "MOD-LEAD", // CANONICAL CODE role: "Ingeniero Delivery" } }
```

For each non-labor estimate:

```
[] { rubroId: "GSV-REU#base_abc123#1", // Unique SK nombre: "Reuniones de seguimiento", category: "Gestión del Servicio", qty: 1, unit_cost: 1000, recurring: true, start_month: 1, end_month: 12, metadata: { source: "baseline", baseline_id: "base_abc123", project_id: "PRJ-X", linea_codigo: "GSV-REU", // CANONICAL CODE } }
```

3. **Rubros stored in DynamoDB**:
   - Table: `rubros`
   - PK: `PROJECT#{projectId}`
   - SK: `RUBRO#{canonicalCode}#{baselineId}#{index}`
   - **CRITICAL**: `metadata.baseline_id` and `metadata.linea_codigo` enable filtering

1.3.5 5. SDMT Forecast Data Loading

Location: `services/finanzas-api/src/handlers/forecast.ts`, `services/finanzas-api/src/`
When SDMT loads forecast data:

1. **Query with baseline filtering** (`GET /plan/forecast?projectId=X`):

```
[] const baselineRubros = await queryProjectRubros(projectId); // This function: // 1. Gets project's active baseline_id from projects.METADATA // 2. Queries all rubros with PK = PROJECT#{projectId} // 3. Filters by metadata.baseline_id === active baseline
```

2. **Generate forecast grid**:
   - For each rubro, create P/F/A cells by month
   - Recurring rubros: spread across start_month to end_month
   - One-time rubros: entire cost in start_month
   - Layer in allocations and payroll actuals

3. **Return to frontend**:

```
[] { "data": [ { "line_item_id": "MOD-LEAD#base_abc123#1", "month": 1, "planned": 7500, "forecast": 7500, "actual": 0 }, ... ], "projectId": "PRJ-X", "months": 12 }
```

1.4 Critical Design Decisions

1.4.1 Why Canonical Taxonomy?

Problem: Hard-coded categories and free-form strings led to: - Mismatches between Estimator and SDMT - Empty forecast data (rubros not found) - Inconsistent role/category listings - No guarantee of MOD lineage

Solution: Single source of truth taxonomy - Frontend uses canonical codes (MOD-LEAD, GSV-REU) - Backend stores linea_codigo in metadata - SDMT views filter by baseline and match on linea_codigo - Guarantees data flows from Estimator → Baseline → SDMT

1.4.2 Why Store linea_codigo in metadata?

The rubroId (SK) must be unique per rubro instance: - `MOD-LEAD#base_abc123#1` - First lead engineer in baseline abc123 - `MOD-LEAD#base_abc123#2` - Second lead engineer in same baseline - `MOD-LEAD#base_xyz789#1` - First lead engineer in different baseline

But we need the **canonical taxonomy code** for: - Grouping multiple instances under same category - Matching with taxonomy definitions - Rolling up costs by linea_codigo - Consistency across baselines

Therefore: `metadata.linea_codigo = "MOD-LEAD"` preserves the canonical link.

1.4.3 Why Filter by baseline_id?

Projects can have multiple baselines over time: - Initial baseline (handed off) - Revised baseline after scope change - Multiple what-if baselines

Without filtering, SDMT would show: - Rubros from all baselines mixed together - Incorrect cost totals (double/triple counting) - Wrong line items in forecast

With filtering: - Only rubros from active baseline shown - Clean handoff between baselines - Accurate cost projections

1.5 Data Flow Example

1.5.1 Step-by-step: Creating "Project Alpha"

1. **PMO creates estimate**:

   - Adds 1x "Ingeniero Delivery" (MOD-LEAD), $6000/mo, 12 months

- Adds 2x "Ingeniero Soporte N2" (MOD-ING), $4000/mo, 12 months
- Adds "Reuniones de seguimiento" (GSV-REU), $1000/mo, 12 months

2. **Submit creates baseline**:
   - baseline_id: `base_e3f6647d3b01`
   - project_id: `PRJ-ALPHA`
   - labor_estimates: `[{ rubroId: "MOD-LEAD", ...}, { rubroId: "MOD-ING", ...}]`
   - non_labor_estimates: `[{ rubroId: "GSV-REU", ...}]`

3. **Handoff materializes rubros**:
   - Rubro 1: `MOD-LEAD#base_e3f6647d3b01#1` → $7500/mo (with on-costs)
   - Rubro 2: `MOD-ING#base_e3f6647d3b01#1` → $5000/mo
   - Rubro 3: `MOD-ING#base_e3f6647d3b01#2` → $5000/mo
   - Rubro 4: `GSV-REU#base_e3f6647d3b01#1` → $1000/mo

4. **Project metadata updated**:

```
[] { pk: "PROJECT#PRJ-ALPHA", sk: "METADATA", baseline_id: "base_e3f6647d3b01", baseline_status: "handed_off" }
```

5. **SDMT loads forecast**:
   - Queries rubros for PRJ-ALPHA
   - Filters where metadata.baseline_id === "base_e3f6647d3b01"
   - Returns 4 rubros × 12 months = 48 forecast cells
   - Total monthly: $18,500 ($7500 + $5000 + $5000 + $1000)

1.6 Testing the Lineage

1.6.1 Manual Verification

1. **Create baseline** via Estimator UI

2. **Check prefacturas table**:

```
[] PK: BASELINE#{baselineId} SK: METADATA // Verify labor_estimates[].rubroId present // Verify non_labor_estimates[].rubroId present
```

3. **Perform handoff** via API

4. **Check projects table**:

```
[] PK: PROJECT#{projectId} SK: METADATA // Verify baseline_id matches
```

5. **Check rubros table**:

```
[] PK: PROJECT#{projectId} SK: RUBRO#* // Verify each rubro has metadata.baseline_id
// Verify metadata.linea_codigo matches taxonomy
```

6. **Load forecast** in SDMT UI

   - Verify non-empty data
   - Verify costs match estimator totals
   - Verify all rubros appear

1.6.2 Automated Tests

Location: `services/finanzas-api/tests/unit/`

- `baseline-sdmt.spec.ts`: Tests baseline filtering logic
- `forecast.spec.ts`: Tests forecast data generation
- `handoff.spec.ts`: Tests handoff and rubro materialization

1.7 Common Issues & Solutions

1.7.1 Issue: Forecast shows zero data

Cause: Rubros not filtered by baseline, or baseline_id mismatch
Solution: - Verify project.baseline_id is set during handoff - Verify each rubro has metadata.baseline_id - Use `queryProjectRubros()` which filters automatically

1.7.2 Issue: Wrong line items in catalog

Cause: Rubros from multiple baselines mixed together
Solution: - Always filter by baseline_id - Use `filterRubrosByBaseline()` helper - Check that SK starts with active baseline_id

1.7.3 Issue: Estimator categories don't match SDMT

Cause: Hard-coded categories in Estimator not aligned with taxonomy
Solution: - Use `useRubrosCatalog()` hook in Estimator - Store canonical rubroId, not display strings - Map roles via `mapModRoleToRubroId()`

1.7.4 Issue: Missing linea_codigo in rubros

Cause: Old baseline data created before taxonomy integration

Solution: - Backfill metadata.linea_codigo based on role/category patterns - Or re-handoff baseline to regenerate rubros with taxonomy

1.8 Future Enhancements

1. **API-based taxonomy**: Fetch rubros dynamically instead of static catalog
2. **Rubro versioning**: Support taxonomy evolution without breaking old baselines
3. **Multi-currency**: Extend lineage to handle FX conversions
4. **Indexation**: Apply CPI/min wage adjustments through lineage
5. **Audit trail**: Track changes to rubros through baseline revisions

1.9 References

- Rubros Taxonomy: `src/modules/rubros.taxonomia.ts`
- MOD Roles: `src/modules/modRoles.ts`
- Baseline Handler: `services/finanzas-api/src/handlers/baseline.ts`
- Handoff Handler: `services/finanzas-api/src/handlers/handoff.ts`
- Projects Handler: `services/finanzas-api/src/handlers/projects.ts`
- Forecast Handler: `services/finanzas-api/src/handlers/forecast.ts`
- Baseline-SDMT Library: `services/finanzas-api/src/lib/baseline-sdmt.ts`