

# **Regression report (last 7 days)**

Finanzas SD – Architecture, Flows & SOPs

Arquitectura, Flujos y Procedimientos

January 8, 2026

## 1 Regression report (last 7 days)

### 1.1 Recent changes touching data flow

- ddc796a6984ec157a8a5b24006ec2923dd640b5c (2025-12-08): Raised /projects limit to 100 and added query param handling without pagination; touched services/finanzas and src/api/finanzas.ts.
- 7597e5bd123ba965071797bb24c8494a53c13551 (2025-12-13): Removed canonical seed tooling (no functional change to listing logic but affects dataset size).
- 7eac0f9 (2025-12-10): Payroll fallback adjustments (unrelated to project listing).

### 1.2 Suspected regression commit

- **Commit:** ddc796a6984ec157a8a5b24006ec2923dd640b5c
- **Reasoning:** Introduced hard limit handling for GET /projects but did not surface pagination tokens or loop through pages. With more than 100 records (now likely after seed removal), end-user-created projects beyond the first page are never returned.
- **Diff summary:** Added limit clamp and request param parsing on the handler and API client (/projects?limit=100), but kept a single DynamoDB ScanCommand without LastEvaluatedKey handling. # Regression Report: SDMT Data Flow Issue

**Date:** 2025-12-13

**Reporter:** Automated Analysis

**Issue:** UI only showing subset of projects (missing end-user-created data)

### 1.3 Commit History (Last 7 Days)

26c3b3f (2025-12-13 21:30:29 +0000) Initial plan

2c5339d (2025-12-13 15:29:31 -0600) Merge pull request #606 from CVDExInfo/codex/i

**Note:** Repository appears to be a shallow clone with grafted history. Only 2 commits visible in recent history.

### 1.4 Suspected Regression Commit

**Commit SHA:** 2c5339d445c625a75115baf808c93a3b2aac965a

**Date:** 2025-12-13 15:29:31 -0600

**PR:** #606

**Title:** "Handle alternative project payload shapes"

**Branch:** codex/investigate-data-lineage-issues

## 1.5 What Changed in PR #606

### 1.5.1 New File Added

**src/api/finanzas-projects-helpers.ts** (NEW) - Introduced `normalizeProjectsPayload()` helper function - Purpose: Handle multiple API response payload shapes - Supports: `data`, `items`, `projects`, `Items`, `results`, `records`, `body.*` variants

### 1.5.2 Impact Analysis

The PR introduced a helper to normalize different API response shapes, but it was **only partially integrated**:

- **Where it IS used:** - `src/modules/finanzas/projects/useProjects.ts` (line 51) - Test file `src/api/__tests__/finanzas.projects.test.ts`

- **Where it IS NOT used:** - `src/lib/api.ts` (`ApiService.getProjects`) - **THE MAIN PROJECT LOADING PATH** - `src/contexts/ProjectContext.tsx` (calls `ApiService.getProjects`)

This created an **inconsistency** where: 1. Some code paths can handle alternate payload shapes (via `normalizeProjectsPayload`) 2. The main project loading path (`ProjectContext`) cannot 3. If backend changes response format or CloudFront/API Gateway transforms it, projects disappear

## 1.6 Reasoning

### 1.6.1 Why This Is The Regression

**Before PR #606:** - All code paths used custom inline payload extraction - Consistent (though limited) handling across the codebase

**After PR #606:** - Introduced a comprehensive normalization helper - But didn't update all code paths to use it - Created a split: some paths robust, some paths fragile - Main project loading path (most critical) wasn't updated

### 1.6.2 The Fragility Introduced

The `ApiService.getProjects()` in `src/lib/api.ts` only checks:

```
[ ] Array.isArray(payload) // Direct array
Array.isArray(payload?.data) // { data: [...] }
Array.isArray(payload?.items) // { items: [...] }
Array.isArray(payload?.data?.items) // { data: { items: [...] } }
```

But `normalizeProjectsPayload()` checks **13 different patterns** including: - `payload.project` - `payload.Items` (DynamoDB) - `payload.results` - `payload.records` - `payload.body.*` variations - Nested variants of all above

### 1.6.3 Why Projects Went Missing

**Scenario 1:** Backend changed response format - If backend started returning { Items: [...] } (DynamoDB style) - Old code path: □ No extraction, falls through to [] - Result: Empty project list

**Scenario 2:** Environment differences - Dev environment returns { data: [...] } □ - Prod environment returns { Items: [...] } □ - Code works in dev, fails in prod

**Scenario 3:** API Gateway transformation - CloudFront or API Gateway wraps response in { body: { data: [...] } } - Extraction logic can't find projects - User sees empty dropdown

## 1.7 Diff Summary of Critical Changes

**File:** src/api/finanzas-projects-helpers.ts (NEW) - Added comprehensive payload normalization function - Handles 13+ different response patterns - Well-tested with 7 test cases

**File:** src/api/finanzas.ts - Exports normalizeProjectsPayload for use by other modules - getProjects() returns response untouched (for downstream normalization)

**File:** src/modules/finanzas/projects/useProjects.ts - □ Updated to use normalizeProject - Now handles all alternate shapes

**File:** src/lib/api.ts - □ **NOT UPDATED** - Still uses limited inline extraction logic - Called by ProjectContext (main project loading path) - **THIS IS THE GAP**

## 1.8 Impact Assessment

### 1.8.1 Affected Components

#### 1. Primary:

- Project dropdown/selector (all SDMT views)
- ProjectContext (global project state)
- Any component calling ApiService.getProjects()

#### 2. Secondary:

- SDMT forecast cards (depend on project selection)
- SDMT metrics/dashboard (depend on project list)
- Project-dependent navigation

### 1.8.2 User Impact

- **Severity:** HIGH (blocks core functionality)
- **Scope:** All users trying to access end-user created projects
- **Symptom:** Empty or partial project list in dropdown

- **Workaround:** None (data is invisible)

### 1.8.3 Why Seed/Demo Projects Might Still Appear

1. **Alphabetical ordering:** Seed projects come first, fit within pagination limit
2. **Different code path:** Seed data might be cached or loaded differently
3. **Response shape:** Seed endpoints might return { data: [...] } which works
4. **Sample size:** If only a few seed projects exist, they all fit in the response

## 1.9 First Bad Commit

**Commit:** 2c5339d445c625a75115baf808c93a3b2aac965a

**Evidence:** 1. Introduced payload normalization helper 2. Did not apply it consistently 3. Created fragility in main project loading path 4. Timing matches symptom appearance

### 1.10 Bisect Not Performed

Due to shallow clone with grafted history, git bisect was not feasible. Analysis based on: - Code review of recent changes - Understanding of data flow - Comparison of extraction logic vs helper logic - Test coverage gaps

### 1.11 Recommended Fix

Apply `normalizeProjectsPayload()` helper in `src/lib/api.ts` at line 152:

```
[ ] // CURRENT (line 150-164) static async getProjects(): Promise<Project[]> { try
{ const payload = await this.request(API_ENDPOINTS.projects);
  logger.info("Projects loaded from API:", payload);
  const projectArray = Array.isArray(payload) ? payload : Array.isArray(payload?.data)
? payload.data : Array.isArray(payload?.items) ? payload.items : Array.isArray(payload?.data?.items)
? payload.data.items : [];
  // RECOMMENDED FIX static async getProjects(): Promise<Project[]> { try { const
payload = await this.request(API_ENDPOINTS.projects);
  logger.info("Projects loaded from API:", payload);
  // Use canonical normalization helper const projectArray = normalizeProjectsPay-
load(payload);
```

This ensures consistent handling across all code paths.

### 1.12 Related Issues

- Data lineage issues mentioned in PR title

- Alternative payload shapes from backend
- CloudFront/API Gateway response transformations
- Environment-specific API formats

---

**Conclusion:** PR #606 introduced a robust payload normalization helper but failed to apply it consistently. The main project loading path was left with fragile extraction logic, causing projects to disappear when response format varies.