

UI network evidence (SDMT cost forecast)

Finanzas SD – Architecture, Flows & SOPs

Arquitectura, Flujos y Procedimientos

January 8, 2026

1 UI network evidence (SDMT cost forecast)

1.1 Endpoints observed

- GET /projects?limit=100 – used by project selector (getProjects in src/api/finanzas.ts)
- GET /baseline?projectId=<id>&project_id=<id> – SDMT baseline payload for metrics/cards.
- GET /allocations?projectId=<id> – cost allocation rows backing SDMT charts.
- GET /payroll?projectId=<id>&project_id=<id> – MOD series for SDMT projections.

1.2 Sample response shapes

- Projects: { data: ProjectDTO[], total: number } or bare array; items carry project_id, código/code, cliente/client, MOD totals, status fields.
- Baseline/Allocations/Payroll: array responses expected by charts (see src/api/finanzas.ts).

1.3 Pagination behavior

- /projects currently requests a single page with limit=100. DynamoDB backend responds with LastEvaluatedKey when more projects exist, but the UI makes no follow-up call.

1.4 Finding

- Live CloudFront endpoint responded with 503 Service Unavailable, so network traces could not be captured directly. Based on API client and handler code, **missing projects are not present in the single-page response** because pagination is not followed; UI does not filter them afterward.

1.5 Before/after comparison

- Before fix: /projects?limit=100 returns first page only; any LastEvaluatedKey is ignored so end-user-created projects beyond the first 100 never reach the UI.
- After fix: backend now scans all pages internally and returns the complete authorized project list in one response (UI still issues a single call). # Evidence: UI Network Analysis - SDMT Data Flow Regression

Date: 2025-12-13

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

Deployed UI: <https://d7t9x3j6j6yd8k.cloudfront.net/finanzas/sdmt/cost/forecast>

1.6 API Endpoints Involved

1.6.1 1. GET /projects (Project Dropdown Population)

URL Path: /projects?limit=100

Query Params: - limit=100 (default)

Request Headers: - Authorization: Bearer <JWT_TOKEN> - Content-Type: application/json

HTTP Status: Expected 200 OK

Response Payload Shape (from backend handler):

```
[] { "data": [ { "projectId": "P-xxxxx", "code": "P-xxxxx", "name": "Project Name", "client": "Client Name", "status": "active", // ... other fields } ], "total": <number> }
```

Backend Handler: services/finanzas-api/src/handlers/projects.ts (lines 1157-1253)

DTO Mapping: services/finanzas-api/src/models/project.ts (mapToProjectDTO function)

Pagination Support: - Backend: DynamoDB ScanCommand with Limit: 100 parameter - No pagination tokens (nextToken/lastEvaluatedKey) in current implementation - UI: Does NOT request additional pages

RBAC Filtering (Backend):

```
[] // Lines 1175-1246 in projects.ts if (userContext.isAdmin || userContext.isExecRO || userContext.isPMO || userContext.isSDMT) { // See all projects - Scan with FilterExpression // sk = METADATA OR sk = META (backward compatibility) } else if (userContext.isSDM) { // Only see projects where sdm_manager_email matches user email } else { // Empty list }
```

1.6.2 2. SDMT Metrics/Cards API Calls

The SDMT forecast view makes additional calls: - GET /plan/forecast?projectId={projectId}&mo

- GET /baseline?projectId={projectId} - GET /allocations?projectId={projectId}

- GET /adjustments?projectId={projectId}

These depend on having a valid projectId selected from the dropdown.

1.7 Issue Analysis

1.7.1 Missing Projects: Present in API vs Filtered in UI?

Hypothesis: Projects are **present in API response** but **dropped during payload normalization** in the frontend.

Evidence:

1. **Backend Query:** Uses ScanCommand with FilterExpression checking for sk = METADATA OR sk = META
 - Seed/canonical projects: Created with sk = "METADATA" ☐
 - End-user projects: Also created with sk = "METADATA" (line 1076 in projects.ts)
 - ☐
 - All projects should pass the filter

2. **Backend Response:** Returns { data: [...], total: N }

- Standard format defined in handler line 1253

3. **Frontend Extraction (src/lib/api.ts lines 156-164):**

```
[] const projectArray = Array.isArray(payload) ? payload : Array.isArray(payload?.data)
? payload.data // Should extract from { data: [...] } : Array.isArray(payload?.items)
? payload.items : Array.isArray(payload?.data?.items) ? payload.data.items : [];
```

This SHOULD work for { data: [...] } format. ☐

4. **BUT:** The frontend extraction logic is **incomplete** compared to normalizeProjectsPayload() helper:

- ☐ Missing support for payload.projects
- ☐ Missing support for payload.Items (DynamoDB style)
- ☐ Missing support for payload.results
- ☐ Missing support for payload.records
- ☐ Missing support for payload.body.* variants

1.7.2 Root Cause Hypothesis

Primary Issue: Frontend payload normalization in src/lib/api.ts (ApiService.getProjects) does NOT use the canonical normalizeProjectsPayload() helper function that was introduced in PR #606.

Impact: - If the API response format changes slightly (e.g., due to CloudFront caching, API Gateway transformations, or backend updates) - Or if different environments return different payload shapes - The frontend extraction logic may fail to extract the project array - Result: Empty or partial project list shown to users

Evidence from PR #606: - A new normalizeProjectsPayload() helper was created to handle multiple response shapes - It's used in src/modules/finanzas/projects/useProject - BUT it's NOT used in src/lib/api.ts (which ProjectContext uses) - This created an inconsistency where some code paths handle alternate payloads, others don't

1.7.3 Secondary Contributors

1. **Pagination Limit:** Backend has a hard limit of 100 projects per request

- If >100 projects exist, not all will be returned
- Frontend does NOT handle pagination tokens
- Could cause missing projects if total > 100

2. **RBAC Filtering:** SDM users only see projects they manage

- If user's role isn't correctly identified
- Or if sdm_manager_email isn't set on projects
- User might see empty list

3. **DynamoDB sk Filtering:** Backward compatibility check for META vs METADATA

- If any projects were created with a different sk value
- They won't be returned by the query

1.8 Why Seeded/Demo Projects Appear But End-User Projects Don't

Current Hypothesis:

1. Seed projects might be created through a different code path that results in a payload shape that the frontend CAN extract
2. End-user projects created via POST /projects might be in a payload format that requires normalizeProjectsPayload() to extract correctly
3. OR: Pagination - seed projects are alphabetically first, so they fit in the limit=100 response

Alternative Hypothesis:

The issue might not be frontend extraction at all - it could be:
- Backend RBAC filtering too aggressive
- DynamoDB query not returning all items (pagination issue on backend)
- Environment/table mismatch (TABLE_PROJECTS pointing to wrong DynamoDB table)
- CloudFront caching returning stale response

1.9 Next Steps

1. Add logging to backend handler to see what's being returned
2. Add logging to frontend API layer to see what's being received
3. Compare frontend extraction logic vs normalizeProjectsPayload logic
4. Test with different user roles (ADMIN vs SDM vs SDMT)
5. Check if pagination is causing truncation
6. Verify DynamoDB table contains end-user created projects

1.10 Before Fix vs After Fix

1.10.1 Before Fix

API Response: (Example with alternate shape)

```
[] { "Items": [ { "projectId": "P-SEED-1", "name": "Seed Project" }, { "projectId": "P-USER-1", "name": "User Project" } ] }
```

Frontend Extraction Result: [] (empty array, because "Items" key not checked)

UI Dropdown: Empty or shows only cached projects

1.10.2 After Fix

API Response: (Same)

```
[] { "Items": [ { "projectId": "P-SEED-1", "name": "Seed Project" }, { "projectId": "P-USER-1", "name": "User Project" } ] }
```

Frontend Extraction Result: [{ projectId: "P-SEED-1", ... }, { projectId: "P-USER-1", ... }]

UI Dropdown: Shows all 2 projects including user-created ones

Conclusion: The issue is most likely in the frontend payload normalization logic not using the canonical normalizeProjectsPayload() helper. This creates an inconsistency where alternate API response shapes are not properly handled.