Fix summary

Finanzas SD – Architecture, Flows & SOPs

Arquitectura, Flujos y Procedimientos

December 17, 2025

1 Fix summary

1.1 Changed files

- `services/finanzas-api/src/handlers/projects.ts` – paginate DynamoDB scans for `/projects`, add structured logs, and return the complete authorized list in one response.
- `services/finanzas-api/tests/unit/projects.rbac.spec.ts` – new coverage proving multi-page scans are stitched together and totals preserved.
- `src/api/__tests__/finanzas.projects.test.ts` – guardrail that large project lists are not truncated on the client normalization path.

1.2 What changed and why

- Replaced single-page scans with a loop over `LastEvaluatedKey`, so all authorized projects are returned even when the table exceeds 100 records.
- Added request/response logs to capture user identity, roles, and pagination state for troubleshooting and auditability.
- Added tests to prevent regressions: backend pagination stitching, contract total preservation, and frontend normalization retaining all items.

1.3 How it resolves the issue

- The UI now receives the full project dataset in its single `/projects` request, so end-user-created projects appear in selectors and downstream SDMT views instead of being dropped after the first DynamoDB page. # Fix Summary: SDMT Data Flow Regression

Date: 2025-12-13
Issue: UI showing only subset of projects (end-user created projects missing)
Root Cause: Incomplete payload normalization in ApiService.getProjects()
Fix Type: Use canonical helper function
Risk: LOW
Lines Changed: 2 (1 import + 1 function call replacement)

1.4 Files Changed

1.4.1 1. `src/lib/api.ts`

Line 30: Added import

[] import { normalizeProjectsPayload } from ”@/api/finanzas-projects-helpers”;

Lines 156-164: Replaced custom extraction logic

```
[] // BEFORE (Limited extraction - only 4 patterns) 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 : [];
// AFTER (Comprehensive extraction - 13+ patterns) const projectArray = normalizeProjectsPayload(payload);
```

1.5 What Changed and Why

1.5.1 The Problem

ApiService.getProjects() is the main path for loading projects in the application, called by: - ProjectContext (src/contexts/ProjectContext.tsx) - All SDMT views that depend on project selection - Project dropdowns and selectors

The function used a **limited** inline extraction logic that only checked 4 response patterns: 1. Direct array: [...] 2. Data wrapper: { data: [...] } 3. Items wrapper: { items: [...] } 4. Nested items: { data: { items: [...] } }

This left it **vulnerable** to: - CloudFront/API Gateway response transformations - Backend format changes - Environment-specific API configurations - Alternative DynamoDB response shapes

1.5.2 The Solution

Replace the inline extraction with the canonical normalizeProjectsPayload() helper that was introduced in PR #606 but not applied consistently.

This helper checks **13+ patterns** including: - payload.data ☐ - payload.items ☐ - payload.projects - payload.Items (DynamoDB) - payload.results - payload.records - payload.body.* (all variants) - payload.data.items ☐ - And more nested combinations

1.5.3 Why This Resolves the Root Cause

1. **Consistency:** Now all code paths use the same normalization logic

   - src/modules/finanzas/projects/useProjects.ts already uses it ☐
   - src/lib/api.ts (ApiService) now uses it ☐
   - No more divergence

2. **Resilience:** Handles all response shapes the backend can return

   - Current format: { data: [...], total: N } ☐
   - DynamoDB format: { Items: [...] } ☐

- Wrapped format: { body: { data: [...] } } ⬜
- Future formats: Automatically supported

3. **Well-Tested:** The helper has 7 comprehensive test cases

- Validates all extraction patterns
- Ensures backward compatibility
- Catches edge cases

4. **Minimal Change:** Only 2 lines modified

- Low risk of introducing new bugs
- Easy to review
- Simple to rollback if needed

1.6 How It Resolves the Root Cause

1.6.1 Before Fix: Data Loss Scenario

```
Backend returns: { body: { data: [P1, P2, P3], total: 3 } }
  ↓
ApiService extraction logic:
  ⬜ Check payload (not array)
  ⬜ Check payload.data → undefined (it's at payload.body.data!)
  ⬜ Check payload.items → undefined
  ⬜ Check payload.data.items → undefined
  ⬜ Falls through to []
  ↓
ProjectContext receives: []
  ↓
UI shows: Empty project dropdown
  ↓
User reports: "My projects are missing!"
```

1.6.2 After Fix: Full Data Extraction

```
Backend returns: { body: { data: [P1, P2, P3], total: 3 } }
  ↓
normalizeProjectsPayload() checks 13+ patterns:
  ⬜ Check payload (not array)
  ⬜ Check payload.data → undefined
  ⬜ Check payload.items → undefined
  ⬜ Check payload.data.items → undefined
  ⬜ Check payload.projects → undefined
```

☐ Check payload.body → found object!
☐ Check payload.body.data → found array with 3 items!
✓ Return [P1, P2, P3]
  ↓
ProjectContext receives: [P1, P2, P3]
  ↓
UI shows: All 3 projects in dropdown
  ↓
User: ☐ Happy, can see all projects

1.7 Testing Strategy

1.7.1 Unit Tests

The normalizeProjectsPayload() function already has comprehensive tests in: - src/api/__tests__/finanzas.projects.test.ts

Tests cover: - ☐ Direct array responses - ☐ Data wrapper: { data: [...] } - ☐ Items wrapper: { items: [...] } - ☐ Nested items: { data: { items: [...] } } - ☐ Projects array: { projects: [...] } - ☐ DynamoDB format: { Items: [...] } - ☐ Body wrapper: { body: { results: [...] } } - ☐ Deeply nested: { body: { data: { projects: [...] } } }
All 7 tests passing ☐

1.7.2 Integration Test

Run existing application:

```
[] npm run build npm run dev
```

Navigate to: - /finanzas/projects (Projects list view) - /finanzas/sdmt/cost/forecast (SDMT forecast view)
Verify: - ☐ Project dropdown populates - ☐ All projects visible (seed + end-user created) - ☐ Project selection works - ☐ SDMT views load data correctly - ☐ No console errors

1.7.3 Regression Test

Test with different API response formats: 1. Standard format: { data: [...], total: N } ☐ 2. DynamoDB format: { Items: [...] } ☐ 3. Wrapped format: { body: { data: [...] } } ☐
All should work correctly now.

1.8 Impact Analysis

1.8.1 Positive Impacts

1. **End-User Projects Visible**

   - Projects created via POST /projects now appear in UI
   - No more "missing projects" reports

2. **Resilience to Format Changes**

   - Backend can change response format
   - Frontend continues to work
   - Reduces coupling

3. **Environment Parity**

   - Dev/stage/prod can have different API Gateway configs
   - Frontend handles all consistently

4. **Future-Proof**

   - New response formats automatically supported
   - Less maintenance burden

1.8.2 Risk Assessment

Change Risk: LOW - Single function change - Uses existing well-tested helper - No logic changes beyond extraction - Backward compatible

   Rollback: EASY - Revert this commit - No database changes - No API contract changes - Immediate effect

   Breaking Changes: NONE - All existing formats still work - New formats now also work - No API changes required

1.9 Deployment Steps

1. **Build & Lint**

   ```
   [] npm run build npm run lint
   ```

2. **Run Tests**

   ```
   [] npm run test:unit
   ```

3. **Deploy to Dev**

```
[] npm run build:finanzas # Deploy dist to S3/CloudFront dev
```

4. **Verify in Dev**

- Login as test user
- Navigate to projects dropdown
- Verify all projects visible
- Test SDMT views

5. **Deploy to Production**

```
[] npm run build:finanzas # Deploy dist to S3/CloudFront prod
```

6. **Monitor**

- Check CloudWatch logs
- Monitor error rates
- Verify user reports decrease

1.10 Guardrails Added

1.10.1 Logging Enhancement

The fix preserves existing logging:

```
[] logger.info("Projects loaded from API:", payload);
```

This logs: - Raw API response (before normalization) - Helps debug future issues - Provides audit trail

1.10.2 Future Recommendations

1. **Add Response Shape Logging**

```
[] logger.info("Extracted projects", { count: projectArray.length, hasData: !!payload.data, hasItems: !!payload.items, hasProjects: !!payload.projects, hasBody: !!payload.body, });
```

2. **Add Backend Response Format Assertion**

- Backend test should assert response shape

- Contract test should validate format
- OpenAPI spec should document structure

3. **Add Frontend Contract Test**

- Test ApiService.getProjects() with various formats
- Ensure normalizeProjectsPayload() called
- Verify extraction works for all shapes

1.11 Related Changes Needed

1.11.1 Backend (Optional Enhancement)

Add structured logging to backend handler:

```
[] // services/finanzas-api/src/handlers/projects.ts line 1253 console.info("[projects]
Returning projects", { count: projects.length, total: projects.length, responseFormat:
"{ data: [...], total: N }", userRole: userContext.roles.join(","), });
```

1.11.2 Documentation (Optional)

Update API contract documentation: - Document expected response format - Note that alternate formats are supported - List all normalized shapes

1.12 Success Metrics

1.12.1 Before Fix

- ☐ Users report: "My projects are missing"
- ☐ Project dropdown: Empty or partial list
- ☐ SDMT views: "No data" or incomplete

1.12.2 After Fix

- ☐ Users report: All projects visible
- ☐ Project dropdown: Full list (seed + user-created)
- ☐ SDMT views: Complete data

1.12.3 KPIs

- **Projects visible:** Should increase from ~10 (seed only) to ~50+ (all)
- **Error rate:** Should decrease
- **User complaints:** Should reduce to zero
- **SDMT usage:** Should increase (now functional)

SUMMARY: This 2-line fix resolves the root cause by using the canonical `normalizeProjectsPay` helper instead of limited inline extraction logic. It makes the frontend resilient to API response format variations while maintaining full backward compatibility.