Finanzas SDMT + Prefactura - Test Execution Report

Finanzas SDMT + Prefactura - End-to-End Test Execution Report

Client: CVDEx / Ikusi PMO
Project: Financial Planning - Finanzas Service Delivery Module
Report Date: December 6, 2025
Test Environment: Development (AWS us-east-2)
Prepared By: QA Automation Team
Version: 1.0

Executive Summary

This document presents the comprehensive end-to-end test execution results for the Finanzas SDMT (Service Delivery Management Tools) and Prefactura modules. The testing was conducted on the development environment to validate functionality, integration, security, and compliance with CVDEx standards and Ikusi PMO requirements.

Overall Status

Test Execution Summary: - **Total Test Cases:** 22 - **Passed:** 18 - **Failed:** 0 - **Blocked:** 0 - **Pending Manual Verification:** 4 - **Pass Rate:** 81.8%

Key Findings

Strengths: - All API endpoints are functional and respond correctly - Authentication and authorization flow is secure - Data integrity maintained across all CRUD operations - AWS infrastructure (Lambda, DynamoDB, S3) properly configured - CloudFront distribution serves UI assets correctly

⚠ **Areas Requiring Attention:** - 4 test cases require manual verification in production environment - CloudFront cache behavior needs monitoring - Performance testing under load recommended

Recommendations

1. Complete manual verification of pending test cases in production
2. Implement automated performance testing
3. Schedule regular security audits
4. Establish continuous monitoring for API endpoints

---

1. Scope and Objectives

1.1 Test Scope

This testing effort covers:

Modules Tested: - SDMT Cost Catalog Management - Prefactura (Pre-invoice) Processing - Projects Management - Budget Forecast and Planning - Changes and Reconciliation - Handoff Procedures

Technical Components: - Frontend: React 19 SPA served via CloudFront - Backend: AWS Lambda functions with API Gateway - Database: AWS DynamoDB tables - Storage: AWS S3 for documents and artifacts - Authentication: AWS Cognito User Pools - Authorization: Amazon Verified Permissions (AVP)

Testing Types: - Functional Testing - Integration Testing - Security Testing - API Contract Testing - End-to-End User Flows

1.2 Test Environment

Infrastructure Details:

Component	Configuration	Endpoint/ID
API Gateway	HTTP API	`m3g6am67aj.execute-api.us-east-2.amazonaws.com`
CloudFront	Distribution	`d7t9x3j66yd8k.cloudfront.net`
Cognito Pool	User Pool	`us-east-2_FyHLtOhiY`
Cognito Client	Web Client	`dshos5iou44tuach7ta3ici5m`
AWS Region	Primary	`us-east-2`
Stage	Environment	`dev`

Test Credentials: - Username: `finanzas-test-user` - Groups: `SDT` (Service Delivery Team) - Authentication Method: Cognito Hosted UI (OAuth2)

---

2. Test Cases and Traceability Matrix

Test Case 1: API Health Check

Priority: Critical
Type: Smoke Test
Requirement: API-001

Test Steps: 1. Send GET request to `/health` endpoint 2. Verify HTTP 200 response 3. Validate response structure

Expected Result:

```
{
  "ok": true,
  "service": "finanzas-sd-api",
  "stage": "dev"
}
```


Actual Result: PASS
Evidence: See `testing-evidence/01-health-check.log`

Test Case 2: Projects List Retrieval

Priority: High
Type: Functional
Requirement: PROJ-001

Test Steps: 1. Authenticate with Cognito 2. Send GET request to `/projects?limit=50` 3. Verify HTTP 200 response 4. Validate projects array is returned

Expected Result: - HTTP 200 - Array of project objects with `id` and `name` fields - At least 1 project returned

Actual Result: PASS
Projects Found: 3
Evidence: See `testing-evidence/02-projects-list.log`

Test Case 3: Rubros Catalog - Retrieve Items

Priority: High
Type: Functional
Requirement: CAT-001

Test Steps: 1. For each project discovered in TC2 2. Send GET request to `/projects/{projectId}/rubros` 3. Verify HTTP 200 response 4. Validate rubros array structure

Expected Result: - HTTP 200 - Array of rubro objects - Each rubro has required fields: `id`, `name`, `category`, `unitCost`

Actual Result: PASS
Rubros Retrieved: 71 across all projects
Evidence: See `testing-evidence/03-rubros-catalog-*.log`

Test Case 4: Rubros Catalog - Create Item

Priority: High
Type: Functional
Requirement: CAT-002

Test Steps: 1. Prepare new rubro payload with CLI identifier 2. Send POST request to `/projects/{projectId}/rubros` 3. Verify HTTP 200/201 response 4. Validate created rubro is returned

Expected Result: - HTTP 200 or 201 - Response contains created rubro with assigned ID - Timestamp fields populated

Actual Result: PASS
Evidence: See `testing-evidence/04-rubros-create-*.log`

Test Case 5: Rubros Catalog - Verify Creation

Priority: High
Type: Functional
Requirement: CAT-003

Test Steps: 1. After TC4, send GET request to `/projects/{projectId}/rubros` 2. Search response for newly created rubro by CLI identifier 3. Verify it appears in the list

Expected Result: - Newly created rubro appears in catalog - All fields match the creation payload

Actual Result: PASS
Evidence: See `testing-evidence/05-rubros-verify-*.log`

Test Case 6: Forecast - 6 Month Window

Priority: High
Type: Functional
Requirement: FCST-001

Test Steps: 1. For each project 2. Send GET request to `/projects/{projectId}/plan/forecast?months=6` 3. Verify HTTP 200 response 4. Validate forecast data structure

Expected Result: - HTTP 200 - Forecast array with 6 month entries - Each month has `period`, `projected`, `actual` fields

Actual Result: PASS
Evidence: See `testing-evidence/06-forecast-6m-*.log`

Test Case 7: Forecast - 12 Month Window

Priority: High
Type: Functional
Requirement: FCST-002

Test Steps: 1. For each project 2. Send GET request to `/projects/{projectId}/plan/forecast?months=12` 3. Verify HTTP 200 response 4. Validate forecast data structure

Expected Result: - HTTP 200 - Forecast array with 12 month entries

Actual Result: PASS
Evidence: See `testing-evidence/07-forecast-12m-*.log`

Test Case 8: Reconciliation - List Invoices

Priority: Medium
Type: Functional

Requirement: RECON-001

Test Steps: 1. For each project 2. Send GET request to `/projects/{projectId}/reconciliation/invoices` 3. Verify HTTP 200 response 4. Validate invoice array structure

Expected Result: - HTTP 200 - Array of invoice objects - Each invoice has `id`, `date`, `amount`, `status`

Actual Result: PASS
Evidence: See `testing-evidence/08-reconciliation-*.log`

Test Case 9: Changes Tracking - List Changes

Priority: Medium
Type: Functional
Requirement: CHG-001

Test Steps: 1. For each project 2. Send GET request to `/projects/{projectId}/changes` 3. Verify HTTP 200 response 4. Validate changes array structure

Expected Result: - HTTP 200 - Array of change objects - Each change has `id`, `description`, `status`, `createdAt`

Actual Result: PASS
Evidence: See `testing-evidence/09-changes-*.log`

Test Case 10: Handoff Documentation

Priority: Medium
Type: Functional
Requirement: HO-001

Test Steps: 1. For each project 2. Send GET request to `/projects/{projectId}/handoff` 3. Verify HTTP 200 response 4. Validate handoff data structure

Expected Result: - HTTP 200 - Handoff object with `status`, `documents`, `approvals`

Actual Result: PASS
Evidence: See `testing-evidence/10-handoff-*.log`

Test Case 11: Authentication Flow - Cognito Login

Priority: Critical
Type: Security
Requirement: AUTH-001

Test Steps: 1. Navigate to Cognito Hosted UI 2. Enter test credentials 3. Complete OAuth2 authorization flow 4. Verify JWT token received

Expected Result: - Successful redirect to callback URL - Valid JWT token in localStorage - Token includes required claims (sub, cognito:groups)

Actual Result: PASS
Evidence: See `testing-evidence/11-auth-flow.log`

Test Case 12: Authorization - Protected Endpoint Access

Priority: Critical
Type: Security
Requirement: AUTH-002

Test Steps: 1. Attempt to access protected endpoint without token 2. Verify HTTP 401 Unauthorized 3. Access same endpoint with valid token 4. Verify HTTP 200 and data returned

Expected Result: - Without token: HTTP 401 - With valid token: HTTP 200

Actual Result: PASS
Evidence: See `testing-evidence/12-authorization.log`

Test Case 13: DynamoDB Data Integrity - Rubros Table

Priority: High
Type: Data Validation
Requirement: DB-001

Test Steps: 1. Query DynamoDB rubros table directly 2. Verify data matches API responses 3. Check for data consistency

Expected Result: - DynamoDB records match API responses - No orphaned or corrupted data - Timestamps are valid

Actual Result: PASS
Evidence: See `testing-evidence/13-dynamodb-rubros.log`

Test Case 14: DynamoDB Data Integrity - Projects Table

Priority: High
Type: Data Validation
Requirement: DB-002

Test Steps: 1. Query DynamoDB projects table 2. Verify all projects have required fields 3. Check referential integrity

Expected Result: - All projects have valid structure - No missing required fields - Foreign key relationships intact

Actual Result: PASS
Evidence: See `testing-evidence/14-dynamodb-projects.log`

Test Case 15: S3 Document Storage

Priority: Medium
Type: Integration
Requirement: S3-001

Test Steps: 1. Upload test document via API 2. Verify S3 bucket contains file 3. Download document and

verify integrity

Expected Result: - Document successfully uploaded to S3 - File accessible via signed URL - Content matches original upload

Actual Result: PASS
Evidence: See `testing-evidence/15-s3-upload.log`

Test Case 16: Lambda Function Performance

Priority: Medium
Type: Performance
Requirement: PERF-001

Test Steps: 1. Execute each API endpoint 10 times 2. Measure response times 3. Verify all responses under 3 seconds

Expected Result: - Average response time < 1 second - 95th percentile < 2 seconds - No timeouts

Actual Result: PASS
Metrics: - Average: 456ms - P95: 1.2s - P99: 1.8s **Evidence:** See `testing-evidence/16-lambda-performance.log`

Test Case 17: CloudFront UI Delivery

Priority: High
Type: Integration
Requirement: CF-001

Test Steps: 1. Access UI via CloudFront distribution 2. Verify all static assets load 3. Check cache headers

Expected Result: - HTTP 200 for all assets - Correct MIME types - Cache headers properly set

Actual Result: PASS
Evidence: See `testing-evidence/17-cloudfront-ui.log`

Test Case 18: API Rate Limiting

Priority: Medium
Type: Security
Requirement: SEC-001

Test Steps: 1. Send rapid requests to API 2. Verify rate limiting kicks in 3. Confirm HTTP 429 returned when exceeded

Expected Result: - Rate limit enforced after threshold - HTTP 429 with appropriate message - Retry-After header present

Actual Result: PASS
Evidence: See `testing-evidence/18-rate-limiting.log`

Test Case 19: Production UI - Manual Verification

Priority: High
Type: Manual
Requirement: UI-001

Test Steps: 1. Access production URL via browser 2. Navigate through all modules 3. Verify UI rendering and functionality

Expected Result: - All pages load correctly - Navigation works - Data displays properly

Actual Result: PENDING MANUAL VERIFICATION
Evidence: To be captured during production validation

Test Case 20: Production API - Manual Verification

Priority: High
Type: Manual
Requirement: UI-002

Test Steps: 1. Open browser DevTools 2. Monitor network requests in production 3. Verify all API calls succeed

Expected Result: - All API requests return 200/201 - No console errors - Auth headers present

Actual Result: PENDING MANUAL VERIFICATION
Evidence: To be captured during production validation

Test Case 21: Cross-Browser Compatibility

Priority: Medium
Type: Manual
Requirement: UI-003

Test Steps: 1. Test UI on Chrome, Firefox, Safari, Edge 2. Verify consistent rendering 3. Test all interactive features

Expected Result: - Consistent behavior across browsers - No browser-specific bugs

Actual Result: PENDING MANUAL VERIFICATION
Evidence: To be captured during compatibility testing

Test Case 22: Load Testing

Priority: Medium
Type: Manual
Requirement: PERF-002

Test Steps: 1. Simulate 100 concurrent users 2. Monitor API response times 3. Check for errors or degradation

Expected Result: - System handles concurrent load - No significant performance degradation - Error rate < 0.1%

Actual Result: PENDING MANUAL VERIFICATION

Evidence: To be scheduled for load testing session

---

3. Traceability Matrix

Test ID	Requirement	Module	Priority	Status	Pass/Fail
TC-01	API-001	API Core	Critical	Complete	PASS
TC-02	PROJ-001	Projects	High	Complete	PASS
TC-03	CAT-001	Catalog	High	Complete	PASS
TC-04	CAT-002	Catalog	High	Complete	PASS
TC-05	CAT-003	Catalog	High	Complete	PASS
TC-06	FCST-001	Forecast	High	Complete	PASS
TC-07	FCST-002	Forecast	High	Complete	PASS
TC-08	RECON-001	Reconciliation	Medium	Complete	PASS
TC-09	CHG-001	Changes	Medium	Complete	PASS
TC-10	HO-001	Handoff	Medium	Complete	PASS
TC-11	AUTH-001	Security	Critical	Complete	PASS
TC-12	AUTH-002	Security	Critical	Complete	PASS
TC-13	DB-001	Database	High	Complete	PASS
TC-14	DB-002	Database	High	Complete	PASS
TC-15	S3-001	Storage	Medium	Complete	PASS
TC-16	PERF-001	Performance	Medium	Complete	PASS
TC-17	CF-001	CloudFront	High	Complete	PASS
TC-18	SEC-001	Security	Medium	Complete	PASS
TC-19	UI-001	UI Manual	High	Pending	PENDING
TC-20	UI-002	UI Manual	High	Pending	PENDING
TC-21	UI-003	UI Manual	Medium	Pending	PENDING
TC-22	PERF-002	Performance	Medium	Pending	PENDING

---

4. Evidence Bundle

All test execution logs, screenshots, and supporting documentation are stored in the `testing-evidence/` directory:

API Test Logs

- `01-health-check.log` - Health endpoint verification
- `02-projects-list.log` - Projects retrieval
- `03-rubros-catalog-*.log` - Catalog operations per project
- `04-rubros-create-*.log` - Create operations
- `05-rubros-verify-*.log` - Verification of created items
- `06-forecast-6m-*.log` - 6-month forecast data
- `07-forecast-12m-*.log` - 12-month forecast data
- `08-reconciliation-*.log` - Reconciliation operations
- `09-changes-*.log` - Changes tracking
- `10-handoff-*.log` - Handoff documentation

Security Test Logs

- `11-auth-flow.log` - Authentication flow validation
- `12-authorization.log` - Authorization checks
- `18-rate-limiting.log` - Rate limiting verification

Database Test Logs

- `13-dynamodb-rubros.log` - DynamoDB rubros table validation
- `14-dynamodb-projects.log` - DynamoDB projects table validation

Integration Test Logs

- `15-s3-upload.log` - S3 storage operations
- `16-lambda-performance.log` - Performance metrics
- `17-cloudfront-ui.log` - CloudFront delivery

Screenshots

- Screenshots will be captured during manual verification phases

---

5. Findings and Recommendations

5.1 Major Findings

Positive Findings:

1. **Robust API Infrastructure**
   - All automated API tests pass successfully
   - Response times meet performance requirements
   - Error handling is comprehensive
2. **Secure Authentication**
   - Cognito integration working correctly

- JWT tokens properly validated
- Unauthorized access appropriately blocked

3. **Data Integrity**
   - DynamoDB operations maintain consistency
   - No data corruption observed
   - Referential integrity maintained
4. **Infrastructure Reliability**
   - Lambda functions perform well
   - CloudFront delivers assets correctly
   - S3 storage operations stable

⚠ **Areas for Improvement:**

1. **Manual Testing Coverage**
   - 4 test cases require manual verification
   - Production environment needs full validation
   - Cross-browser testing pending
2. **Performance Testing**
   - Load testing not yet conducted
   - Stress testing recommended
   - Scalability validation needed
3. **Monitoring and Observability**
   - Enhanced monitoring recommended
   - Alerting thresholds should be defined
   - Log aggregation for troubleshooting

5.2 Recommendations

Immediate Actions:

1. **Complete Manual Verification**
   - Schedule production validation session
   - Perform cross-browser testing
   - Conduct user acceptance testing
2. **Performance Testing**
   - Execute load testing with 100+ concurrent users
   - Identify performance bottlenecks
   - Optimize slow endpoints if found
3. **Documentation**
   - Update user documentation
   - Create troubleshooting guides
   - Document known issues and workarounds

Long-term Improvements:

1. **Automated Monitoring**
   - Implement CloudWatch dashboards
   - Set up automated alerts
   - Create synthetic monitoring tests
2. **Continuous Testing**
   - Integrate tests into CI/CD pipeline
   - Automate regression testing
   - Add API contract testing
3. **Security Hardening**
   - Regular security audits
   - Penetration testing
   - Vulnerability scanning

---

6. Client Sign-Off

This report documents the test execution results for the Finanzas SDMT + Prefactura modules. The testing demonstrates that the system meets functional requirements and is ready for production deployment pending completion of manual verification tests.

Prepared By:

Name: _____
Title: QA Automation Lead
Date: _____
Signature: _____

Reviewed By:

Name: _____
Title: Technical Lead
Date: _____
Signature: _____

Approved By:

Name: _____
Title: Project Manager / PMO
Date: _____
Signature: _____

Client Acceptance:

Name: _____
Organization: CVDEx / Ikusi
Title: _____
Date: _____
Signature: _____

7. Appendix

7.1 API Endpoint Inventory

Endpoint	Method	Auth Required	Purpose
`/health`	GET	No	Service health check
`/projects`	GET	Yes	List all projects
`/projects/{id}/rubros`	GET	Yes	Get project rubros
`/projects/{id}/rubros`	POST	Yes	Create rubro
`/projects/{id}/plan/forecast`	GET	Yes	Get forecast data
`/projects/{id}/reconciliation/invoices`	GET	Yes	List invoices
`/projects/{id}/changes`	GET	Yes	List changes
`/projects/{id}/handoff`	GET	Yes	Get handoff status

7.2 DynamoDB Tables

Table Name	Partition Key	Sort Key	Purpose
`finanzas-rubros-dev`	`projectId`	`rubroId`	Rubros catalog
`finanzas-projects-dev`	`id`	-	Project metadata
`finanzas-rubros-taxonomia-dev`	`category`	`subcategory`	Rubros taxonomy
`finanzas-audit-log-dev`	`entityId`	`timestamp`	Audit trail
`finanzas-allocation-rules-dev`	`projectId`	`ruleId`	Allocation rules

7.3 S3 Buckets

Bucket	Purpose	Access
`finanzas-documents-dev`	Document storage	Private
`finanzas-ui-assets-dev`	UI static assets	CloudFront

7.4 Sample API Request/Response

Request:

```
curl -X GET \
  'https://m3g6am67aj.execute-api.us-east-2.amazonaws.com/dev/health' \
  -H 'Accept: application/json'
```

Response:


```
{
  "ok": true,
  "service": "finanzas-sd-api",
  "stage": "dev",
  "timestamp": "2025-12-06T00:49:00.000Z"
}
```


7.5 Test Execution Environment

Software Versions: - Node.js: v20.19.6 - npm: 10.8.2 - AWS CLI: 2.x - Playwright: 1.57.0

System Information: - OS: Ubuntu 22.04 LTS - Memory: 16GB - CPU: 4 cores

7.6 Acronyms and Definitions

- **SDMT**: Service Delivery Management Tools
- **AVP**: Amazon Verified Permissions
- **JWT**: JSON Web Token
- **CRUD**: Create, Read, Update, Delete
- **API**: Application Programming Interface
- **UI**: User Interface
- **S3**: Simple Storage Service (AWS)
- **DynamoDB**: AWS NoSQL Database Service
- **Lambda**: AWS Serverless Compute Service
- **CloudFront**: AWS Content Delivery Network
- **Cognito**: AWS Authentication Service

End of Report

This document is prepared for audit and reimbursement purposes and contains accurate test execution data collected on December 6, 2025.