Finanzas API Deployment Guide

Finanzas SD – Architecture, Flows & SOPs

Arquitectura, Flujos y Procedimientos

January 8, 2026

1 Finanzas API Deployment Guide

1.1 Overview

The Finanzas SD API uses a **single environment deployment model** for the current pilot phase. This simplifies infrastructure management and reduces operational complexity.

1.2 Current Deployment Configuration

1.2.1 Single Environment Approach

The API is deployed using:

- **Stack Name**: `finanzas-sd-api-dev`
- **Stage**: `dev`
- **Table Prefix**: `finz_`
- **API ID**: `m3g6am67aj`
- **API URL**: `https://m3g6am67aj.execute-api.us-east-2.amazonaws.com/dev`
- **Region**: `us-east-2`

1.2.2 Why Single Environment?

During the pilot phase, we use a single environment to: - Reduce infrastructure costs - Simplify deployment workflows - Minimize configuration drift - Accelerate iteration cycles - Focus on feature development rather than environment management

1.3 Deployment Workflow

The API is deployed automatically via GitHub Actions workflow: `.github/workflows/deploy-api.ym`

1.3.1 Trigger Conditions

The workflow triggers on pushes to: - `module/finanzas-api-mvp - r1-finanzas-dev-wiring - main`

Or manually via `workflow_dispatch`.

1.3.2 Deployment Steps

1. **Preflight Checks**: Validates all required environment variables
2. **AWS Authentication**: Uses OIDC for secure AWS access
3. **AVP Verification**: Optionally verifies Amazon Verified Permissions policy store
4. **Build**: Compiles TypeScript and packages Lambda functions
5. **Deploy**: Uses AWS SAM to deploy the stack

6. **Guard Checks**: Validates API ID and required routes
7. **Smoke Tests**: Tests public and protected endpoints
8. **Seed Data**: Populates DynamoDB with catalog data

1.3.3 Key Parameters

All deployments use these fixed values: - `StageName=dev` - `TablePrefix=finz_` - `FINZ_API_STACK=finanzas-sd-api-dev`

1.4 DynamoDB Tables

All tables use the `finz_` prefix:

- `finz_projects` - Project metadata
- `finz_rubros` - Budget line items catalog
- `finz_rubros_taxonomia` - Budget taxonomy
- `finz_allocations` - Resource allocations
- `finz_payroll_actuals` - Payroll actuals
- `finz_adjustments` - Budget adjustments
- `finz_alerts` - System alerts
- `finz_providers` - Provider information
- `finz_audit_log` - Audit trail

1.5 API Routes

1.5.1 Public Endpoints (No Authentication Required)

- `GET /health` - Health check
- `GET /catalog/rubros` - Budget line items catalog

1.5.2 Protected Endpoints (JWT Required)

- `GET /allocation-rules` - Allocation rules
- `POST /projects` - Create project
- `GET /adjustments` - Budget adjustments
- And more...

1.6 Authentication

The API uses **Cognito JWT authentication** with: - **Authorizer**: CognitoJwt - **User Pool ID**: Configured via `COGNITO_USER_POOL_ID` variable - **Client ID**: Configured via `COGNITO_WEB_CLIENT` variable

1.7 Guard Checks

The deployment workflow enforces: 1. **API ID Stability**: Ensures the API ID remains `m3g6am67aj` 2. **Required Routes**: Verifies mandatory routes exist 3. **Authorizer Presence**: Confirms CognitoJwt authorizer is configured

These guards prevent accidental API recreation or misconfiguration.

1.8 UI Integration

The Finanzas UI integrates with this API using the base URL constructed from: - `FINZ_API_ID_DEV` = `m3g6am67aj` - Stage = `dev` - Region = `us-east-2`

Result: `https://m3g6am67aj.execute-api.us-east-2.amazonaws.com/dev`

1.9 Future Considerations

When transitioning from pilot to production: - Consider creating separate `prod` environment - Implement blue-green deployment strategy - Add comprehensive monitoring and alerting - Implement automated rollback capabilities - Document environment-specific configurations

1.10 Troubleshooting

1.10.1 Deployment Failures

If deployment fails: 1. Check CloudWatch logs for Lambda functions 2. Verify all required GitHub secrets and variables are set 3. Review SAM deployment output for specific errors 4. Ensure DynamoDB tables exist and are accessible

1.10.2 API Not Responding

If API is unresponsive: 1. Check `/health` endpoint first 2. Verify Lambda function CloudWatch logs 3. Check API Gateway configuration 4. Verify Cognito configuration for protected endpoints

1.10.3 Guard Check Failures

If guard checks fail: 1. Verify API ID matches expected value (`m3g6am67aj`) 2. Check that all required routes are defined in the SAM template 3. Ensure CognitoJwt authorizer is properly configured

1.11 Additional Resources

- SAM Template
- Deploy API Workflow

- API Source Code