

Repository Structure Documentation

Finanzas SD – Architecture, Flows & SOPs

Arquitectura, Flujos y Procedimientos

November 11, 2025

1 Repository Structure Documentation

Last Updated: November 9, 2025

Module: Finanzas (R1) - Financial Planning & Service Delivery

Status: Pre-production, AWS SDK integration active

1.1 Overview

This repository contains a multi-module enterprise financial planning application with:

- **SDMT Module:** Service delivery cost tracking and forecasting
- **Finanzas Module (R1):** Budget management (Rubros catalog, allocation rules)

The application is built with React 19, Vite 6, TypeScript, and Tailwind CSS v4, deployed behind CloudFront at /finanzas/.

1.2 Directory Structure

```

/
  └── src/
      ├── api/                                # Application source code
          └── finanzasClient.ts                # Real AWS API clients
      ├── components/                         # Finanzas module AWS SDK client
          ├── ui/                               # Shared UI components
              ├──AuthProvider.tsx             # Shadcn/Radix UI primitives
              ├──Navigation.tsx              # Authentication context
              ├──Protected.tsx               # Main navigation bar
              └──...                            # Route protection HOC
          ├── config/                         # Other shared components
              └── aws.ts                      # Configuration files
          ├── contexts/                      # AWS Cognito/API configuration
              └── ProjectContext.tsx        # React contexts
          ├── features/                      # Project selection state
              ├── HomePage.tsx              # Feature modules (domain-driven)
              ├── pmo/                        # Landing page
              └── sdmr/                       # PMO features
                  └── cost/                 # SDMT features
                      ├── Catalog/           # Cost management
                      ├── Forecast/          # Line items catalog
                      └── Reconciliation/    # Budget forecasting

```

```

|   |           └── Cashflow/      # Cash flow analysis
|   |           └── Scenarios/    # What-if scenarios
|   |               └── Changes/    # Change requests
|   ├── hooks/                                # Custom React hooks
|   └── lib/
|       ├── api.ts                            # Mock API service (PMO/SDMT)
|       ├── auth.ts                           # Auth utilities
|       └── utils.ts                          # General utilities
|
|   └── mocks/                               # Mock data for development
|       ├── baseline*.json                  # Project baselines (3 projects)
|       ├── billing-plan*.json            # Billing schedules
|       ├── forecast*.json                # Forecast data
|       ├── invoices*.json                # Invoice samples
|       └── ikusi-service-catalog.json     # Service catalog
|
|   └── modules/                            # Independent feature modules
|       └── finanzas/                      # Finanzas R1 module
|           ├── FinanzasHome.tsx          # Finanzas landing
|           ├── RubrosCatalog.tsx        # Budget items catalog
|           ├── AllocationRulesPreview.tsx # Allocation rules
|           └── data/                   # Static Finanzas data
|
|   └── types/                                # TypeScript type definitions
|       └── domain.d.ts                  # Domain models
|
|   └── App.tsx                                # Root application component
|
|   └── main.tsx                                # Application entry point
|
└── docs/                                     # Documentation
    ├── archive/                             # Historical implementation docs
    ├── adr/                                 # Architecture decision records
    ├── architecture/                        # System architecture docs
    ├── ops/                                 # Operations guides
    ├── runbooks/                            # Operational runbooks
    ├── tree.structure.md                    # This file
    └── *.md                                 # Current documentation
|
└── infra/                                    # Infrastructure as code
    └── terraform/                         # Terraform configurations
|
└── scripts/                                 # Build and deployment scripts
    └── create-s3-bucket.sh                 # S3 bucket creation

```

```

|   └── generate-docs-pdf.cjs      # PDF documentation generator

|   └── services/                 # Backend services
|       └── finanzas-api/         # Finanzas API (AWS SAM)

|   └── openapi/                  # API specifications
|       └── finanzas.yaml        # Finanzas API contract

|   └── public/                   # Static assets
|   └── .github/                  # GitHub workflows and configs
|   └── package.json              # Dependencies and scripts
|   └── vite.config.ts            # Vite build configuration
|   └── tsconfig.json             # TypeScript configuration
|   └── tailwind.config.js        # Tailwind CSS configuration

```

1.3 Key Architecture Decisions

1.3.1 Data Layer Strategy

The application uses **dual data access patterns** based on module:

PMO/SDMT Modules (Mock-based)

- **File:** src/lib/api.ts
- **Purpose:** Mock API service for PMO and SDMT features
- **Data Source:** JSON files in src/mocks/
- **Usage:** Development and testing of PMO/SDMT workflows
- **Toggle:** VITE_USE_MOCK_API=true/false (default: true in dev)
- **Rationale:** PMO/SDMT backend is not yet implemented; mocks enable UI development

Finanzas Module (AWS SDK-based)

- **File:** src/api/finanzasClient.ts
- **Purpose:** Real AWS API client for Finanzas module
- **Data Source:** AWS API Gateway → Lambda → DynamoDB
- **Endpoint:** https://m3g6am67aj.execute-api.us-east-2.amazonaws.com/dev
- **Auth:** JWT token in localStorage (finz_jwt)
- **Rationale:** Finanzas R1 has live backend infrastructure

1.3.2 Module Separation

No centralization needed - The current structure already provides clean separation:

- PMO/SDMT features use mock data layer (`lib/api.ts`) - Finanzas module uses real AWS SDK client (`api/finanzasClient.ts`)
- Each module imports from its appropriate data source

1.3.3 Feature Organization

Features are organized by **business domain** under `src/features/`:

- `pmo/` - Project Management Office features
- `sdmt/` - Service Delivery Management Team features

- Finanzas is a standalone module under `src/modules/finanzas/`

This structure supports:

- Clear ownership boundaries
- Independent deployment (via `BUILD_TARGET` env var)
- Gradual backend integration

1.4 Mock Data Strategy

1.4.1 Purpose

Mock data enables frontend development while backend services are under construction.

1.4.2 Active Mocks (Retained)

All mock files in `src/mocks/` are **actively used** by PMO/SDMT features:

- `baseline*.json` (3 variants: healthcare, fintech, retail)
- `billing-plan*.json` (3 variants)
- `forecast*.json` (3 variants)
- `invoices*.json` (3 variants)
- `ikusi-service-catalog.json` (service tier definitions)

1.4.3 Usage Pattern

```
[ ] // src/lib/api.ts imports mocks import baselineData from '@/mocks/baseline.json';
import serviceCatalog from '@/mocks/ikusi-service-catalog.json';
// Components use ApiService import ApiService from '@/lib/api'; const projects =
await ApiService.getProjects();
```

1.4.4 Future Migration

When PMO/SDMT backend is ready:

1. Set `VITE_USE_MOCK_API=false` in production
2. Update `src/lib/api.ts` to call real endpoints
3. Keep mocks for local development and testing

1.5 Removed Components (Cleanup)

1.5.1 Development-Only Components (Removed)

- ☐ src/pages/_diag/FinanzasDiag.tsx - Diagnostics dashboard
- ☐ src/components/RoleDebugPanel.tsx - Role switching panel
- ☐ Route /_diag - Diagnostic route

Rationale: These were dev-time debugging tools not needed in production deployment.

1.5.2 Documentation Cleanup

- ☐ Moved 55+ historical implementation docs to docs/archive/
- ☐ Retained essential docs in root: README, PRD, SECURITY, LICENSE
- ☐ Kept docs/ structure for architectural and operational documentation

1.6 Build Configuration

1.6.1 Build Targets

The application supports multiple build targets via BUILD_TARGET environment variable:

```
[] # Finanzas-only build (default) BUILD_TARGET=finanzas npm run build  
# PMO-only build (future) BUILD_TARGET=pmo npm run build
```

1.6.2 Base Path Configuration

- **Vite base:** /finanzas/ (configured in vite.config.ts)
- **Router basename:** /finanzas (configured in App.tsx)
- **CloudFront path:** /finanzas/* (behavior pattern)
- **S3 prefix:** Objects stored under finanzas/ key prefix

This ensures: - Deep links work correctly (e.g., /finanzas/catalog/rubros) - Assets load properly behind CloudFront - SPA routing functions as expected

1.7 Environment Variables

1.7.1 Core Settings

```
[ ] # API Configuration VITE_API_BASE_URL=https://m3g6am67aj.execute-api.us-east-2.amazonaws.com/dev VITE_USE_MOCK_API=false # Use real API (true for local dev with mocks) VITE_FINZ_ENABLED=true # Enable Finanzas module
# Authentication VITE_SKIP_AUTH=true # Skip auth for development VITE_COGNITO_POOL_ID=us-east-2_FyHLtOhiY VITE_COGNITO_CLIENT_ID=dshos5iou44tuach7ta3ici5m
# Deployment VITE_APP_BASENAME=/finanzas
```

See `.env.production` for full production configuration.

1.8 Testing Strategy

1.8.1 Current State

- No unit test framework installed (Vitest planned but not configured)
- Manual verification via:
 - Build success (`npm run build`)
 - Lint pass (`npm run lint`)
 - Local preview (`npm run preview`)
 - Manual UI testing

1.8.2 Test Infrastructure (Future)

- `src/_tests_/basePath.test.ts` - Contains commented-out tests for Vitest
 - When Vitest is added, uncomment tests and add to CI/CD
-

1.9 Deployment Pipeline

1.9.1 CI/CD

- **Workflow:** `.github/workflows/deploy-ui.yml`
- **Trigger:** Push to main branch
- **Steps:**
 1. Build with `BUILD_TARGET=finanzas`
 2. Sync to S3 bucket ukusi-ui-finanzas-prod
 3. Invalidate CloudFront distribution EPQU7PVDLQXUA

1.9.2 Manual Deployment

```
[ ] npm run build:finanzas aws s3 sync dist/ s3://ukusi-ui-finanzas-prod/finanzas/ --  
delete aws cloudfront create-invalidation --distribution-id EPQU7PVDLQXUA --paths '/fi-  
nanzas/*'
```

1.10 Routing Structure

1.10.1 Public Routes

- / - Home page (FinanzasHome if VITE_FINZ_ENABLED=true, else HomePage)
- /profile - User profile

1.10.2 PMO Routes

1.10.3 SDMT Routes

- /sdmt/cost/catalog - Line items catalog
- /sdmt/cost/forecast - Budget forecasting
- /sdmt/cost/reconciliation - Invoice reconciliation
- /sdmt/cost/cashflow - Cash flow analysis
- /sdmt/cost/scenarios - What-if scenarios
- /sdmt/cost/changes - Change requests

1.10.4 Finanzas Routes

- /catalog/rubros - Rubros (budget items) catalog
- /rules - Allocation rules preview

1.11 Security Considerations

1.11.1 Authentication

- Cognito User Pool: us-east-2_FyHLt0hiY
- JWT tokens stored in localStorage (finz_jwt)
- Auth can be skipped in dev via VITE_SKIP_AUTH=true

1.11.2 Authorization

- Role-based access control (RBAC) implemented at UI level
- Roles: PMO, SDMT, VENDOR, EXEC_RO
- Backend authorization handled by Lambda authorizers (not in UI scope)

1.11.3 Data Protection

- S3 bucket is private (no public access)
 - CloudFront + OAC (Origin Access Control)
 - HTTPS enforced
 - No secrets in source code (environment variables only)
-

1.12 Next Steps for Finanzas R2

1. Backend Integration

- Complete PMO/SDMT API implementation
- Replace mock data layer with real API calls
- Add retry logic and error handling

2. Testing

- Install and configure Vitest
- Add unit tests for components and utilities
- Add integration tests for critical workflows

3. Performance

- Implement code splitting (dynamic imports)
- Optimize bundle size (currently 2.1MB)
- Add lazy loading for routes

4. Monitoring

- Add CloudWatch RUM for frontend monitoring
 - Implement error tracking (Sentry or similar)
 - Add performance metrics
-

1.13 Questions & Support

- **Repository:** valencia94/financial-planning-u
- **Branch:** copilot/cleanup-finanzas-module
- **Documentation:** See docs/ directory for detailed guides
- **Issues:** GitHub Issues for bug reports and feature requests