CVDex

QA-Component-Analysis

Finanzas SD – Architecture, Flows & SOPs

Arquitectura, Flujos y Procedimientos

November 10, 2025

1 Finanzas UI/API Component Analysis

Date: 2025-11-09
Purpose: Detailed analysis of Finanzas module implementation for QA review

---

1.1 Component Inventory

1.1.1 1. UI Components

FinanzasHome.tsx

- **Location:** `src/modules/finanzas/FinanzasHome.tsx`
- **Route:** `/` (with basename `/finanzas/`)
- **Purpose:** Landing page for Finanzas module
- **Features:**
  - Static page with heading and description
  - Two action cards for navigation
  - Links to Rubros Catalog and Allocation Rules
- **API Calls:** None (static page)
- **Status:** ✅ Implemented
- **Test Priority:** High

Implementation Details:

```
- Displays heading: "Finanzas · Gestión Presupuesto (R1)"
- Two cards:
  1. "Catálogo de Rubros" → /catalog/rubros
  2. "Reglas de Asignación" → /rules
- Simple navigation-focused design
```

RubrosCatalog.tsx

- **Location:** `src/modules/finanzas/RubrosCatalog.tsx`
- **Route:** `/catalog/rubros`
- **Purpose:** Display catalog of budget line items (rubros)
- **Features:**
  - Fetches rubros from API
  - Displays data in table format
  - Shows loading state
  - Error handling

- **API Calls:** GET `/catalog/rubros`
- **Status:** ✅ Implemented
- **Test Priority:** Critical

Implementation Details:

```
- Uses finanzasClient.getRubros()
- Displays table with columns:
  - rubro_id
  - nombre
  - categoria
  - linea_codigo
  - tipo_costo
- Loading state: "Cargando…"
- Error state: displays error message in red
- Empty state: "No hay rubros disponibles."
- Footer: Shows count of rubros
```

Data Flow:

1. Component mounts
2. useEffect triggers API call via finanzasClient
3. Loading state displayed
4. On success: setRows(data)
5. On error: setError(message)
6. Table renders with data

AllocationRulesPreview.tsx

- **Location:** `src/modules/finanzas/AllocationRulesPreview.tsx`
- **Route:** `/rules`
- **Purpose:** Display allocation rules for budget distribution
- **Features:**
  - Fetches rules from API
  - Displays rules in card format
  - Shows active/inactive status
  - Displays split information
- **API Calls:** GET `/allocation-rules`
- **Status:** ✅ Implemented
- **Test Priority:** High

Implementation Details:

```
- Direct fetch call (not using finanzasClient)
- API endpoint: ${VITE_API_BASE_URL}/allocation-rules
- No auth header sent (potential issue?)
- Displays rules with:
  - rule_id
  - linea_codigo
  - driver
  - active status badge
  - split details (if present)
  - fixed_amount (if present)
- Loading state: "Loading allocation rules..."
- Error state: "Error: {message}"
- Empty state: "No rules found."
```

Data Flow:

1. Component mounts
2. useEffect triggers direct fetch
3. Loading state displayed
4. On success: setRules(json.data || [])
5. On error: setError(message)
6. Cards render with data

1.2 API Client Analysis

1.2.1 finanzasClient.ts

- **Location:** `src/api/finanzasClient.ts`
- **Purpose:** Centralized API client for Finanzas endpoints
- **Status:** □ Implemented, partially used

Configuration:

- Base URL: `VITE_API_BASE_URL` (env variable)
- Auth: Bearer token from localStorage or env
- Token storage: `localStorage.getItem('finz_jwt')`
- Fallback: `VITE_API_JWT_TOKEN` (static test token)

Methods:

1. `health()` - GET /health
2. `getRubros()` - GET /catalog/rubros (with schema validation)

Observations:

- Rubros endpoint uses Zod schema validation
- Health endpoint implemented
- No method for allocation-rules (AllocationRulesPreview uses direct fetch)
- Token stored in localStorage under key 'finz_jwt'

Recommendations:

1. Add getAllocationRules() method to finanzasClient
2. Update AllocationRulesPreview to use finanzasClient
3. Add schema validation for allocation rules

1.3 Routing Configuration

1.3.1 App.tsx Routes

- **Basename:** `/finanzas/` (when VITE_FINZ_ENABLED=true)
- **Feature Flag:** `VITE_FINZ_ENABLED`

Route Mapping:

```
/ → FinanzasHome (when VITE_FINZ_ENABLED=true)
/catalog/rubros → RubrosCatalog
/rules → AllocationRulesPreview
```

Access Control:

- Wrapped in component
- Requires authentication
- Navigation only shows when authenticated

1.4 API Endpoints Inventory

1.4.1 Implemented in Backend (per docs)

1. GET /health (public)
2. GET /catalog/rubros (authenticated)

3. □ GET /allocation-rules (authenticated)
4. ⚠ GET /projects (may be 501 not implemented)
5. ⚠ POST /projects (may be 501 not implemented)
6. ⚠ GET /adjustments (may be 501 not implemented)
7. ⚠ POST /adjustments (may be 501 not implemented)
8. ⚠ GET /movements (may be 501 not implemented)
9. ⚠ POST /payroll/ingest (may be 501 not implemented)

1.4.2 Used by UI

- □ GET /health (finanzasClient.health)
- □ GET /catalog/rubros (RubrosCatalog)
- □ GET /allocation-rules (AllocationRulesPreview)

1.4.3 Not Yet Used by UI

- GET /projects
- POST /projects
- GET /adjustments
- POST /adjustments
- GET /movements
- POST /payroll/ingest
- Close Month workflow
- Prefactura workflow

1.5 Authentication Flow

1.5.1 Expected Flow (per documentation)

1. User signs in via Cognito
2. Cognito returns ID token (JWT)
3. Token stored in localStorage as 'finz_jwt'
4. Token sent in Authorization header: `Bearer {token}`
5. API validates token via Cognito JWT authorizer

1.5.2 Current Implementation

- finanzasClient checks localStorage for 'finz_jwt'
- Falls back to VITE_API_JWT_TOKEN env variable
- AllocationRulesPreview does NOT send auth header (bug?)

1.5.3 Issues Identified

1. □ AllocationRulesPreview missing auth header
2. ⚠ No visible sign-in flow in UI components
3. ⚠ Token storage mechanism not clear
4. ⚠ Token refresh not implemented

1.6 Missing Features / Not Implemented

1.6.1 Charts & Dashboards

- □ No chart components found
- □ No dashboard visualizations
- □ No budget allocation charts
- □ No cost breakdown visualizations

1.6.2 Reports & Export

- □ No print functionality
- □ No export to Excel
- □ No export to PDF
- □ No save/download buttons

1.6.3 CRUD Operations

- □ No create forms
- □ No edit forms
- □ No delete operations
- □ No bulk operations

1.6.4 Workflows

- □ Close Month not implemented
- □ Prefactura approval not implemented
- □ Payroll ingest not implemented
- □ Project handoff not implemented

1.6.5 Search & Filter

- □ No search functionality in Rubros
- □ No filter options
- □ No sorting capabilities

- □ No pagination

1.6.6 UI Enhancements

- ⚠ Loading states basic but functional
- ⚠ Error handling basic but functional
- □ No success notifications
- □ No validation messages

1.7 Test Coverage Assessment

1.7.1 What Can Be Tested (Implemented)

1. □ Sign-in flow (external, via Cognito)
2. □ Navigation to Finanzas home
3. □ Navigation to Rubros catalog
4. □ Navigation to Rules preview
5. □ API call to GET /health
6. □ API call to GET /catalog/rubros
7. □ API call to GET /allocation-rules
8. □ Loading states display
9. □ Error states display
10. □ Data rendering in tables/cards

1.7.2 What Cannot Be Tested (Not Implemented)

1. □ Charts rendering
2. □ Dashboard functionality
3. □ Print/export features
4. □ CRUD operations
5. □ Workflows (Close Month, etc.)
6. □ Search/filter/sort
7. □ Pagination
8. □ Form validation

1.8 Current State Summary

1.8.1 □ Working Features

- Basic navigation structure

- Home page with action cards
- Rubros catalog display with API integration
- Allocation rules display with API integration
- Loading and error states
- Table-based data display
- API client with schema validation (rubros)

1.8.2 ⚠ Partially Working / Issues

- AllocationRulesPreview missing auth header
- No finanzasClient method for allocation rules
- Token management not fully visible
- No schema validation for rules

1.8.3 □ Not Implemented

- Charts and dashboards
- Reports and export functionality
- CRUD operations
- Workflow operations
- Search, filter, sort
- Pagination
- Advanced error handling
- Success notifications

1.9 Recommendations

1.9.1 Critical (Must Fix Before Go-Live)

1. Add auth header to AllocationRulesPreview
2. Add getAllocationRules() to finanzasClient
3. Add schema validation for allocation rules
4. Verify token storage and retrieval mechanism
5. Test unauthorized access handling

1.9.2 High Priority (Should Have)

1. Add basic search functionality to Rubros
2. Add sorting to Rubros table
3. Add proper loading indicators
4. Improve error messages

5. Add success notifications

1.9.3 Medium Priority (Nice to Have)

1. Add pagination to Rubros
2. Add filters for categories/types
3. Add export to Excel functionality
4. Add print preview
5. Implement basic charts

1.9.4 Future Enhancements

1. CRUD operations for rubros
2. CRUD operations for rules
3. Workflow implementations
4. Advanced dashboards
5. Real-time updates

1.10 Test Strategy

1.10.1 Phase 1: Core Functionality (Current Sprint)

- Test authentication flow
- Test navigation
- Test API calls (health, rubros, rules)
- Test data display
- Test error handling
- Fix AllocationRulesPreview auth issue

1.10.2 Phase 2: Enhanced Features (Future Sprint)

- Implement and test search/filter
- Implement and test sorting
- Implement and test pagination
- Implement and test export features

1.10.3 Phase 3: Advanced Features (Future Sprint)

- Implement and test CRUD operations
- Implement and test workflows
- Implement and test charts/dashboards

1.11 Conclusion

The Finanzas module has a solid foundation with:

- Working navigation
- Basic data display for rubros and rules
- API integration via finanzasClient
- Proper error handling

However, it is a minimal MVP with:

- No charts or dashboards
- No reports or export features
- No CRUD operations
- No workflows
- Limited search/filter capabilities

For go-live readiness, we need to:

1. Fix the auth header issue in AllocationRulesPreview
2. Complete thorough testing of existing features
3. Document limitations clearly
4. Plan for future enhancements

The current implementation is suitable for an R1 MVP that demonstrates the concept and allows users to view catalog data and rules.