# TDZ Fix: Finanzas SD Cost Catalog Route

Finanzas SD – Architecture, Flows & SOPs

Arquitectura, Flujos y Procedimientos

January 23, 2026

# 1 TDZ Fix: Finanzas SD Cost Catalog Route

## 1.1 Issue Summary

**Route:** `/finanzas/sdmt/cost/catalog`
**Error:** `ReferenceError: Cannot access 'B' before initialization`
**Component:** `src/features/sdmt/cost/Catalog/SDMTCatalog.tsx`
**Date Fixed:** 2025-11-17

## 1.2 Root Cause

The component had a **Temporal Dead Zone (TDZ) violation** where a variable was referenced before its declaration:

```
[] // □ BEFORE (lines 83-87): Using lineItemsError BEFORE declaration const lineIt-
emsErrorMessage = lineItemsError ? lineItemsError instanceof Error ? lineItemsEr-
ror.message : String(lineItemsError) : "";
// Line 114: lineItemsError declared here (TOO LATE!) const { error: lineItemsError,
... } = useProjectLineItems();
```

## 1.3 Solution

Moved the `useProjectLineItems()` hook call to execute **before** any usage of its returned values:

```
[] // □ AFTER: Hook called first (lines 105-111) const { lineItems: queryLineItems,
isLoading: isLineItemsLoading, isFetching: isLineItemsFetching, error: lineItemsError,
invalidate: invalidateLineItems, } = useProjectLineItems();
// Then compute dependent values (lines 113-118) const lineItemsErrorMessage =
lineItemsError ? lineItemsError instanceof Error ? lineItemsError.message : String(lineItemsError)
: "";
```

## 1.4 Why This Happened

JavaScript/TypeScript's const and let declarations have **block scope** with a TDZ, meaning: - The variable exists in the scope from the start - But it cannot be accessed until the line where it's declared and initialized - Accessing it before declaration throws `ReferenceError: Cannot access 'X' before initialization`

In minified code, variable names are shortened (e.g., `lineItemsError` → B), which is why the error message referenced variable B.

### 1.5 Prevention Guidelines

To avoid similar TDZ issues in React components:

1. **Declare all hooks at the top** of the component function, following React's Rules of Hooks
2. **Order declarations properly:**
   - State hooks (`useState`, etc.)
   - Context hooks (`useContext`, custom hooks like `useProject()`)
   - Computed values that depend on hook results
3. **Never reference a variable before its declaration line**
4. **Be careful with destructuring** - ensure the source is declared first
5. **Watch for circular dependencies** between modules (though not the issue here)

### 1.6 Files Modified

- `src/features/sdmt/cost/Catalog/SDMTCatalog.tsx`: Reordered hook calls and variable declarations

### 1.7 Verification

- □ Build succeeds without errors
- □ TypeScript type-checking passes (no new errors)
- □ ESLint passes with no warnings
- □ Route renders without runtime errors

---

## 2 TDZ Fix: Finanzas SD Cost Forecast Route - "Cuadrícula de Pronóstico 12 Meses"

### 2.1 Issue Summary

**Route:** `/finanzas/sdmt/cost/forecast`
**Error:** `ReferenceError: Cannot access 'X' before initialization`
**Component:** `src/features/sdmt/cost/Forecast/components/ForecastRubrosTable.tsx`
**Date Fixed:** 2026-01-11
**Trigger:** Clicking "Cuadrícula de Pronóstico 12 Meses" (12-Month Forecast Grid) to expand the collapsible section

## 2.2 Root Cause

The `ForecastRubrosTable` component had a **Temporal Dead Zone (TDZ) violation** where a function was called before its declaration:

```
[] // □ BEFORE: Function called inside useMemo (line 192) const visibleCategories
= useMemo(() => { // ... filtering logic ... const recalculatedTotals = recalculate-
CategoryTotals(filteredRubros); // □ Called here // ... }, [searchFilter, categoryTotals,
categoryRubros, filterMode]);
// Line 201: Function declared here (TOO LATE!) const recalculateCategoryTotals
= (rubros: CategoryRubro[]): CategoryTotals => { // ... implementation ... };
```

When the `visibleCategories` useMemo executes, it tries to call `recalculateCategoryTotals`, but that function hasn't been initialized yet because it's declared later in the code.

In minified production builds, `recalculateCategoryTotals` gets shortened to a single letter (e.g., X), causing the error:

```
ReferenceError: Cannot access 'X' before initialization
```

## 2.3 Solution

Moved the `recalculateCategoryTotals` function declaration **before** the `visibleCategories` useMemo:

```
[] // □ AFTER: Function declared first (line 159) const recalculateCategoryTotals =
(rubros: CategoryRubro[]): CategoryTotals => { // ... implementation ... };
// Then used in useMemo (line 203+) const visibleCategories = useMemo(() => { //
... filtering logic ... const recalculatedTotals = recalculateCategoryTotals(filteredRubros);
// □ Now safe // ... }, [searchFilter, categoryTotals, categoryRubros, filterMode]);
```

## 2.4 Why This Happened

JavaScript/TypeScript const and let declarations are hoisted but remain in the **Temporal Dead Zone (TDZ)** until the execution reaches the declaration line:

1. The variable name is reserved in scope from the start
2. But it cannot be accessed until the line where it's declared and initialized
3. Accessing it before declaration throws `ReferenceError: Cannot access 'X' before initialization`
4. In production builds, variable names are minified (e.g., `recalculateCategoryTotals` → X)

The issue only appeared when expanding the “Cuadrícula de Pronóstico 12 Meses” section because: - That’s when `ForecastRubrosTable` component is first rendered - The `visibleCategories` useMemo executes immediately on mount - It tries to call `recalculateCategoryTotals` which is still in TDZ

### 2.5 Prevention Guidelines

To avoid similar TDZ issues in React components:

1. **Declare helper functions BEFORE they’re used** in useMemo/useEffect/useCallback
2. **Order declarations properly:**
   - Imports
   - Type definitions
   - Helper functions (that don’t depend on state/props)
   - State hooks (`useState`, etc.)
   - Context hooks (`useContext`, custom hooks)
   - Computed values (useMemo, useCallback) that use the helpers
3. **Never reference a const/let before its declaration line**
4. **Consider moving complex helper functions outside the component** if they don’t need closure over state/props
5. **Use ESLint rules** to catch potential TDZ issues during development

### 2.6 Files Modified

- `src/features/sdmt/cost/Forecast/components/ForecastRubrosTable.tsx`: Moved `recalculateCategoryTotals` function declaration before `visibleCategories` useMemo
- `docs/notes-finanzas-tdz.md`: Updated documentation with this case

### 2.7 Verification

- □ Build succeeds without errors
- □ TypeScript type-checking passes
- □ ESLint passes with no warnings
- □ Route loads without errors
- □ Clicking “Cuadrícula de Pronóstico 12 Meses” renders without TDZ error
- □ Component can be dynamically imported without throwing

## 3 TDZ Prevention: Comprehensive Codebase Hardening (2026-01-22)

### 3.1 Issue Summary

**Scope:** Repository-wide preventive TDZ hardening
**Date Fixed:** 2026-01-22
**Trigger:** Proactive security and stability improvement

### 3.2 Root Cause

While no active TDZ errors were manifesting in production, the codebase lacked systematic protection against Temporal Dead Zone issues. Without ESLint enforcement, developers could accidentally introduce code where:

1. `const`/`let` arrow functions are used before definition
2. Classes are instantiated before declaration
3. Variables are referenced before initialization
4. Imports are placed after usage (incorrect ordering)

These issues only surface at runtime (especially in minified bundles), making them hard to debug.

### 3.3 Solution

#### 3.3.1 1. Added ESLint Rule for TDZ Prevention

Added `@typescript-eslint/no-use-before-define` rule to `eslint.config.js`:

```
[] '@typescript-eslint/no-use-before-define': ['error', { functions: false, // Function declarations are hoisted, so allow them classes: true, // Class declarations must be defined before use variables: true, // Variables (const/let) must be defined before use allowNamedExports: false }]
```

This rule catches TDZ violations at **development time** rather than runtime.

#### 3.3.2 2. Fixed All Existing TDZ Violations (36 total)

**Core API and Data Modules (5 fixes)**

- **`src/lib/rubros/canonical-taxonomy.ts`**: Moved `normalizeKey` import to top of file
- **`src/api/finanzas.ts`**:
  - Hoisted `FinanzasApiError` class before `fetchJson`
  - Hoisted `validateArrayResponse` helper before first usage
  - Hoisted `applyTaxonomy` helper before `normalizeLineItem`

**Backend Handlers (5 fixes)**

- **`services/finanzas-api/src/handlers/baseline.ts`**:
  - Moved `getBaseline` before route handler
  - Moved `getProjectBaseline` before route handler
  - Moved `getBaselineRubros` before route handler
  - Moved `getBaselineAllocations` before route handler
  - Moved `getBaselinePrefacturas` before route handler

**Frontend Components (26 fixes across 13 files)**

- **`src/api/finanzasClient.ts`**: Moved `ProjectRubroRequestSchema` before usage
- **`src/components/ImportWizard.tsx`**: Reordered 4 helper functions
- **`src/components/baseline/BaselineStatusPanel.tsx`**: Moved `normalizedStatus` computation
- **`src/components/budget/AnnualBudgetWidget.tsx`**: Moved `loadBudget` before useEffect
- **`src/components/charts/EnhancedCharts.tsx`**: Moved `formatValue` helper
- **`src/components/ui/chart.tsx`**: Moved `ChartStyle` definition
- **`src/components/ui/form.tsx`**: Moved `FormItemContext` creation
- **`src/features/pmo/prefactura/Estimator/steps/FXIndexationStep.tsx`**: Moved `getHedgingDescription`
- **`src/features/sdmt/cost/Cashflow/SDMTCashflow.tsx`**: Moved `loadCashflowData`
- **`src/features/sdmt/cost/Forecast/SDMTForecast.tsx`**: Reordered 2 load functions
- **`src/features/sdmt/cost/Forecast/components/hooks/useMonthlySnapshotData.ts`**: Reordered 6 helper functions
- **`src/features/sdmt/cost/Forecast/useSDMTForecastData.ts`**: Moved `normalizeString` and status computation
- **`src/features/sdmt/cost/Reconciliation/SDMTReconciliation.tsx`**: Moved `formatRubroLabel`
- **`src/features/sdmt/cost/Scenarios/SDMTScenarios.tsx`**: Moved `loadScenarios`

## 3.4 Why This Matters

### 3.4.1 Before: Silent Runtime Bombs

```
[] // □ DANGEROUS: Works in dev, fails in production minified build const data = useMemo(() => transformData(items), [items]); const transformData = (items) => { … }; // Defined AFTER usage
```

In development, this might work due to bundler quirks, but in production: - Variables get minified to single letters - Execution order becomes critical - Error: `ReferenceError: Cannot access 'X' before initialization`

### 3.4.2 After: Compile-Time Safety

```
[] // □ SAFE: ESLint enforces correct order const transformData = (items) => { … }; // Defined FIRST const data = useMemo(() => transformData(items), [items]);
```

ESLint catches the issue during development, preventing production bugs.

## 3.5 Prevention Guidelines for Developers

### 3.5.1 1. Module-Level Code Order

```
[] // □ CORRECT ORDER import { someHelper } from './utils'; // 1. Imports first
export class MyClass { … } // 2. Classes
export const CONSTANT = 'value'; // 3. Constants
// Helper functions function helperA() { … } // 4. Functions (hoisted, but good practice) const helperB = () => { … }; // 5. Arrow functions BEFORE usage
// Module-level code const data = helperB(); // 6. Code using helpers
```

### 3.5.2 2. React Component Order

```
[] // □ CORRECT ORDER inside component function MyComponent() { // 1. Helper functions FIRST const transformData = (items) => { … }; const formatValue = (val) => { … };
// 2. State hooks const [data, setData] = useState([]);
// 3. Context/custom hooks const project = useProject();
// 4. Computed values using helpers const transformed = useMemo(() => transformData(data), [data]); const formatted = useMemo(() => formatValue(transformed), [transformed]);
// 5. Effects useEffect(() => { … }, []);
// 6. Event handlers const handleClick = () => { … };
// 7. Render return <div>…</div>; }
```

### 3.5.3 3. Avoid Common Pitfalls

```
[] // □ WRONG: Using before defining const result = calculate(data); const calculate
= (d) => d * 2;
// □ WRONG: Importing after usage const x = normalizeKey(input); import { nor-
malizeKey } from './utils';
// □ WRONG: Class before definition const instance = new MyClass(); class MyClass
{ ... }
```

## 3.6 Verification

- □ **36 TDZ violations fixed** (0 remaining)
- □ **ESLint passes** with no errors
- □ **Build succeeds** without warnings
- □ **Production bundle** tested (no runtime TDZ errors)
- □ **TypeScript compilation** passes
- □ **No logic changes** - only declaration reordering

## 3.7 Files Modified

**Total: 18 files**

1. `eslint.config.js` - Added TDZ prevention rule
2. `src/lib/rubros/canonical-taxonomy.ts`
3. `src/api/finanzas.ts`
4. `services/finanzas-api/src/handlers/baseline.ts`
5. `src/api/finanzasClient.ts`
6. `src/components/ImportWizard.tsx`
7. `src/components/baseline/BaselineStatusPanel.tsx`
8. `src/components/budget/AnnualBudgetWidget.tsx`
9. `src/components/charts/EnhancedCharts.tsx`
10. `src/components/ui/chart.tsx`
11. `src/components/ui/form.tsx`
12. `src/features/pmo/prefactura/Estimator/steps/FXIndexationStep.tsx`
13. `src/features/sdmt/cost/Cashflow/SDMTCashflow.tsx`
14. `src/features/sdmt/cost/Forecast/SDMTForecast.tsx`
15. `src/features/sdmt/cost/Forecast/components/hooks/useMonthlySnapshotData.ts`
16. `src/features/sdmt/cost/Forecast/useSDMTForecastData.ts`
17. `src/features/sdmt/cost/Reconciliation/SDMTReconciliation.tsx`
18. `src/features/sdmt/cost/Scenarios/SDMTScenarios.tsx`

### 3.8 Future Protection

The ESLint rule now **prevents** new TDZ issues from being introduced:

```
[] gitaddsrc/my−component.tsx pnpm run lint
□ Error: 'helper' was used before it was defined → Fix: Move 'helper' definition before
line 42
```

Developers get immediate feedback, preventing TDZ bugs from reaching production.