

# Informe Tecnico: normalizer.ts

REPORTE TECNICO: normalizer.ts

## 1. PROPOSITO Y FUNCION PRINCIPAL

El archivo "src/utils/normalizer.ts" actua como la Capa de Adaptacion ("Shim Layer") del Frontend. Su funcion vital es actuar de "puente" entre la estructura cruda de la base de datos (MongoDB/Firestore V3) y los componentes visuales de la aplicacion (React), que historicamente esperaban un formato diferente ("Legacy").

Problema que resuelve:

La base de datos moderna ("funds\_v3") tiene datos anidados (derived.asset\_class, std\_perf.volatility), mientras que muchos componentes de la UI esperan propiedades planas. Sin este archivo, la aplicacion fallaria.

## 2. FUNCIONES CLAVE

A. adaptFundV3ToLegacy(docData)

Objetivo: Transformacion estructural "burda" para compatibilidad inmediata.

Toma el objeto crudo de V3 y "aplana" los campos mas criticos.

- Entrada: Objeto V3

- Salida: Objeto hibrido con alias legacy.

B. normalizeFundData(docDataInput)

Objetivo: Estandarizacion estricta y calculo de metricas finales.

Transformaciones Criticas:

1. Metricas de Rendimiento:

- Busca en "std\_perf" (Backend).
- Si falla, busca en "ms.risk\_volatility" (Morningstar V3).
- Si falla, busca en "perf" (Legacy).
- Resultado: Garantiza volatilidad y sharpe numericos.

2. Metricas de Porcentaje:

- Detecta si es base 100 (8.5) o decimal (0.085) y estandariza a decimal.

3. Objeto "std\_perf\_norm":

- Crea un sub-objeto limpio para la UI moderna (X-Ray, etc).

3. CAMBIOS RECIENTES (Enero 2026)

Hemos modificado este archivo para soportar "Direct V3 Search":

- Antes: Si no existia std\_perf, mostraba 0.

- Ahora: Lee "ms.risk\_volatility" para asegurar que siempre haya datos, incluso si el backend no ha recalculado hoy.

## CONCLUSION

"normalizer.ts" es el traductor universal del proyecto. Asegura que el Frontend hable el mismo idioma que el Backend.