Design System Visual Documentation

Finanzas SD – Architecture, Flows & SOPs

Arquitectura, Flujos y Procedimientos

January 29, 2026

1 Design System Visual Documentation

This document provides visual examples and usage patterns for the Tier 3 design system components.

1.1 DashboardLayout

The `DashboardLayout` component provides a 12-column responsive grid system.

1.1.1 Basic Usage

```
[] import { DashboardLayout, DashboardSection } from '@/components/ui/design-system';
function MyDashboard() { return ( <DashboardLayout maxWidth="2xl"> <DashboardSection colSpan={12}> {/* Full width content */} </DashboardSection>
<DashboardSection colSpan={6}> {/* Half width - left side */} </DashboardSection> <DashboardSection colSpan={6}> {/* Half width - right side */} </DashboardSection> </DashboardLayout> ); }
```

1.1.2 Responsive Layout

```
[] <DashboardLayout> <DashboardSection colSpan={12} responsive={{ md: 6, lg: 4 }} > {/* Mobile: full width, Tablet: 6 cols, Desktop: 4 cols */} </DashboardSection> </DashboardLayout>
```

1.1.3 Helper Components

```
[] import { DashboardHalf, // 6 columns (responsive) DashboardThird, // 4 columns (responsive) DashboardQuarter // 3 columns (responsive) } from '@/components/ui/design-system';
<DashboardLayout> <DashboardHalf>{/* Content */}</DashboardHalf> <DashboardHalf>{/* Content */}</DashboardHalf> </DashboardLayout>
```

1.2 StandardTable

Enhanced table with density controls and sticky headers.

1.2.1 Basic Usage

```
[] import {  StandardTable,  StandardTableHeader,  StandardTableHead,  Standard-
TableBody,  StandardTableRow,  StandardTableCell } from '@/components/ui/design-
system';
function MyTable() {  return (  <StandardTable  density="comfortable"  sticky-
Header={true} showDensityToggle={true} > <StandardTableHeader> <tr> <Stan-
dardTableHead>Column 1</StandardTableHead> <StandardTableHead>Column 2</Stan-
dardTableHead> </tr> </StandardTableHeader> <StandardTableBody> <Standard-
TableRow> <StandardTableCell>Data 1</StandardTableCell> <StandardTableCell>Data
2</StandardTableCell>  </StandardTableRow>  </StandardTableBody>  </Standard-
Table>  ); }
```

1.2.2 Density Options

- **Compact**: Smaller padding (px-2 py-1.5)
- **Comfortable**: Default padding (px-3 py-3)

1.3 Chip Components

1.3.1 Status Chips

Color-coded chips for success/warning/error states with auto-icons.

```
[] import { StatusChip } from '@/components/ui/design-system';
<StatusChip status="success">Completed</StatusChip> <StatusChip status="warning">Pend
<StatusChip status="error">Failed</StatusChip> <StatusChip status="info">In Progress</StatusC
```

Visual Guide: - □ Success: Green background with checkmark icon - □ Warning: Yellow background with alert icon - □ Error: Red background with alert icon - □ Info: Blue background with info icon

1.3.2 Category Chips

Neutral grey chips for categorization.

```
[] import { CategoryChip } from '@/components/ui/design-system';
<CategoryChip>MOD</CategoryChip> <CategoryChip>OPEX</CategoryChip> <Cat-
egoryChip>CAPEX</CategoryChip>
```

1.3.3 Tag Chips

Outlined style for lightweight labels.

```
[] import { TagChip } from '@/components/ui/design-system';
<TagChip>En Meta</TagChip> <TagChip>Sobre presupuesto</TagChip>
```

1.3.4 VarianceChip

Specialized chip for financial variances with automatic threshold colors.

```
[] import { VarianceChip } from '@/components/ui/design-system';
// Automatic variant selection based on thresholds <VarianceChip variance={1000}
percentage={5} formatValue={(v) => formatCurrency(v)} warningThreshold={10}
// Yellow at 10% errorThreshold={20} // Red at 20% />
```

Threshold Logic: - variance = 0 → Success (green) - |percentage| < 10% → Info (blue) - 10% ≤ |percentage| < 20% → Warning (yellow) - |percentage| ≥ 20% → Error (red)

1.4 Theme Tokens

1.4.1 Spacing Scale

```
[] import { spacing } from '@/lib/design-system/theme';
// spacing.xs = '0.5rem' // 8px // spacing.sm = '0.75rem' // 12px // spacing.md =
'1rem' // 16px // spacing.lg = '1.5rem' // 24px // spacing.xl = '2rem' // 32px // spac-
ing['2xl'] = '3rem' // 48px
```

1.4.2 Grid Configuration

```
[] import { grid } from '@/lib/design-system/theme';
// grid.columns = 12 // grid.gutter = '1rem' // 16px // grid.margin = '1.5rem' // 24px
```

1.4.3 Color Palette

```
[] import { colors } from '@/lib/design-system/theme';
// Status colors colors.status.success // Green colors.status.warning // Yellow col-
ors.status.error // Red colors.status.info // Blue
// Neutral scale (50-900) colors.neutral[50] // Lightest colors.neutral[500] // Middle
colors.neutral[900] // Darkest
```

1.5 Taxonomy Helpers

Utility functions for working with rubros taxonomy.

```
[] import { getFuenteReferenciaLabel, getAllCategories, searchTaxonomy, get-
CategoryColor } from '@/lib/rubros/taxonomyHelpers';
// Map codes to user-friendly labels const label = getFuenteReferenciaLabel('PMO');
// ”PMO” const label2 = getFuenteReferenciaLabel('SAP'); // ”ERP / Contabilidad (SAP)”
// Get all categories const categories = getAllCategories(); // Returns: [{ codigo:
'QLT', nombre: 'Calidad y Mejora Continua' }, ...]
// Search taxonomy const results = searchTaxonomy('ISO'); // Returns matching
RubroTaxonomyItem[]
// Get color for category const color = getCategoryColor('PMO'); // ”purple” const
color2 = getCategoryColor('QLT'); // ”emerald”
```

1.6 Feature Flag

All design system components are behind the `VITE_FINZ_NEW_DESIGN_SYSTEM` feature flag.

```
[] # Enable new design system VITE_FINZ_NEW_DESIGN_SYSTEM=true npm run dev
# Disable (default) VITE_FINZ_NEW_DESIGN_SYSTEM=false npm run dev
```

1.7 Migration Guide

1.7.1 Migrating to DashboardLayout

Before:

```
[] <div className=”max-w-7xl mx-auto p-6”> <div className=”grid grid-cols-12
gap-4”> <div className=”col-span-6”>Content</div> </div> </div>
```

After:

```
[] <DashboardLayout maxWidth=”2xl”> <DashboardSection colSpan={6}>Content</Dashboa
</DashboardLayout>
```

1.7.2 Migrating to VarianceChip

Before (old VarianceChip):

```
[] <VarianceChip value={1000} percent={15} ariaLabel="Budget variance" />
```

After (new design system VarianceChip):

```
[] <VarianceChip variance={1000} percentage={15} formatValue={(v) => formatCurrency(v)} />
```

1.8 Browser Compatibility

- Modern browsers (Chrome, Firefox, Safari, Edge)
- CSS Grid support required
- CSS custom properties support required

1.9 Accessibility

All components follow WCAG 2.1 AA guidelines: - Color contrast ratios meet minimum requirements - All interactive elements have focus states - ARIA labels provided where appropriate - Semantic HTML structure