Clase 14 – Style Guides y Design Systems

IIC2182 - Interfaces y Experiencia de Usuario



PONTIFICIA
UNIVERSIDAD
CATÓLICA
DE CHILE

Primer Semestre 2026

Agenda

De la inconsistencia caótica a un lenguaje compartido entre diseño y código.

El hilo

- El problema: consistencia
- Style Guide
- Design System

Componentes

- 9 piezas esenciales
- Design Tokens
- Atomic Design

Estrategia

- Adoptar / Adaptar / Crear
- Ejemplos del mundo real
- Caso práctico: shadcn/ui

Hoy hablamos de la infraestructura de diseño que permite escalar productos sin perder consistencia.

Agenda

De la inconsistencia caótica a un lenguaje compartido entre diseño y código.

El hilo

- El problema: consistencia
- Style Guide
- Design System

Componentes

- 9 piezas esenciales
- Design Tokens
- Atomic Design

Estrategia

- Adoptar / Adaptar / Crear
- Ejemplos del mundo real
- Caso práctico: shadcn/ui

Hoy hablamos de la infraestructura de diseño que permite escalar productos sin perder consistencia.

Sin sistema, todo diverge

Cuatro síntomas que aparecen siempre que un producto crece sin un lenguaje común.

1. Inconsistencia visual

Cada pantalla usa un azul distinto.

Botones con padding y radios diferentes en cada vista.

El usuario nota algo raro — pero no sabe qué.

2. Reinventar la rueda

Cinco versiones del mismo Modal, cada una con un bug diferente.

El equipo gasta tiempo resolviendo lo mismo.

3. Fricción diseño ↔ código

El diseñador entrega un mock; el dev lo interpreta a su manera.

Las revisiones se vuelven una negociación de pixeles.

4. Onboarding lento

Una persona nueva no sabe qué componente reutilizar.

Sin documentación, el conocimiento vive en cabezas.

La solución no son más reuniones. Es un artefacto compartido. Hoy vemos dos: Style Guide y Design System.

Style Guides

La primera solución al problema de la consistencia.

¿Qué es un Style Guide?

Definición

Un documento que define el lenguaje visual y verbal de una marca: cómo se ve, cómo suena y cómo se aplica de forma consistente.

Foco en branding y aplicación
Tipografía, color, logo, voz

Formato simple
Un PDF, una página web o un Notion bien armado

Lo escribe alguien — no se 'instala'
Una persona puede mantenerlo

Para muchos productos pequeños, un Style Guide bien hecho es todo lo que necesitan.

Anatomía mínima de un Style Guide

Seis bloques que aparecen en casi todas las marcas serias — el mínimo viable.

Color

Paleta primaria, secundaria, neutros y semánticos (success, warning, error).

Reglas de uso, contraste mínimo, fondos válidos.

Aa Tipografía

Familias, jerarquía (H1–H6, body, caption), pesos y line-height.

Type scale: 12 / 14 / 16 / 20 / 24 / 32 / 48.

Logo

Variantes (full, mark, mono), espacio mínimo, fondos válidos.

Lo que no se puede hacer es tan importante como lo que sí.

Voz y tono

Cómo "suena" la marca al escribir.

Cercana vs formal, técnica vs casual — siempre con ejemplos.

Iconografía e imagen

Estilo de íconos (lineal, sólido, duotono), tratamiento de fotografías, ilustración.

Espaciado y layout

Grilla base, escala de espacios (4 / 8 / 16 / 24…), márgenes.

El puente natural hacia los Design Tokens.

Style Guides reales para inspirarse

Recursos públicos que vale la pena revisar antes de armar el tuyo.

BBC GEL

Global Experience Language. Manual público de la BBC — referencia clásica de cómo documentar tipografía, color y voz.

Mailchimp Content Style Guide

El estándar de oro para voz, tono y reglas de redacción. Solo texto — pero define la marca entera.

Material You — Color

Sistema dinámico de color generado desde una sola "key color". Estado del arte en tokens de color.

Apple HIG — Foundations

Capítulos de typography, color y materials. Fuertemente prescriptivo — útil para entender qué nivel de detalle exige una marca seria.

¿Cuándo el Style Guide ya no alcanza?

Un Style Guide cubre el qué se ve. Cuando además quieres definir el cómo se construye, necesitas un Design System.

Style Guide alcanza cuando

Tienes un solo producto.

El equipo es pequeño y se comunica de forma directa.

El frontend lo escribe una persona o dos.

Los cambios de marca son raros.

Necesitas un Design System cuando

Hay varios productos compartiendo la misma marca.

Diseñadores y devs trabajan en paralelo en muchos features.

Quieres garantizar accesibilidad de forma sistemática.

El producto debe vivir años con coherencia.

Un Design System incluye un Style Guide — y le agrega componentes, tokens, código y documentación.

¿Qué es un Design System?

Empecemos por dos definiciones canónicas.

Definiciones

Nielsen Norman Group

"Un conjunto completo de estándares pensados para gestionar el diseño a escala mediante componentes y patrones reutilizables."

Interaction Design Foundation

"Un conjunto de estándares para gestionar y escalar el diseño, que incorpora componentes reutilizables, principios y guías para mantener la consistencia y agilizar el flujo de trabajo."

La palabra clave es escala. Un Design System tiene sentido cuando hay varios productos, varios equipos y varios años de trabajo por delante.

Design System ≠ Style Guide

Términos que se confunden, pero tienen alcances muy distintos.

Design System

Alcance: integral

Incluye: tokens, componentes, patrones, guías, código, accesibilidad

Foco: elementos funcionales y visuales

Propósito: experiencia unificada entre productos

Style Guide

Alcance: reducido

Incluye: tipografía, color, logo, tono de voz

Foco: lenguaje visual y branding

Propósito: consistencia de marca

Un Style Guide es un **subconjunto** de un Design System. Todo Design System contiene una guía visual — pero un Style Guide solo no es un Design System.

Las 9 piezas de un Design System

Lo que la IxDF identifica como componentes esenciales.

Las 9 piezas

Tres capas: fundamentos, biblioteca y directrices.

1 Brand Values

Los principios que definen quiénes somos como producto.

2 Design Principles

Reglas que guían las decisiones de diseño.

3 Brand Style Guide

Tipografía, color, logo — el lenguaje visual de la marca.

4 Design Tokens

Variables nombradas: `color-primary`, `spacing-md`.

5 Component Library

Botones, inputs, modales — UI elements reutilizables.

6 Pattern Library

Soluciones a problemas recurrentes: formularios, tablas, navegación.

7 Icon Library

Símbolos visuales con guías de uso y tamaño.

8 Content Guidelines

Tono, estilo y reglas de redacción del producto.

9 Accessibility

Cómo asegurar uso por todas las personas (WCAG).

No todos los Design Systems implementan las 9 piezas — pero las maduras sí.

Design Tokens

El alma del Design System: una capa de variables compartidas.

¿Qué es un Design Token?

Un nombre para un valor de diseño — color, espaciado, tipografía, sombra, radio.

```
// tokens.json
{
  "color": {
    "primary":   "#0EA5E9",
    "secondary": "#8B5CF6",
    "danger":    "#EF4444",
    "text":      "#111827"
  },
  "spacing": {
    "xs": "4px",
    "sm": "8px",
    "md": "16px",
    "lg": "24px",
    "xl": "48px"
  },
  "radius": {
    "sm": "4px",
    "md": "8px",
    "lg": "16px"
  }
}
```

Una sola fuente de verdad

Diseñadores y desarrolladores referencian los mismos nombres.

Cambio en cascada

Cambiar `color-primary` propaga la actualización a todo el sistema.

Multi-plataforma

Los mismos tokens se exportan a CSS, iOS, Android, Figma.

Niveles de tokens

Buenas prácticas: organizar tokens en tres capas con propósitos distintos.

1. Global / Primitive

Valores crudos, sin contexto.

```
blue-500: #0EA5E9
```

```
spacing-2: 8px
```

2. Semantic / Alias

Significado de uso.

```
color-action-primary → blue-500
```

```
color-text-danger → red-600
```

3. Component

Específico al componente.

```
button-primary-bg → color-action-primary
```

```
button-padding-x → spacing-md
```

Atomic Design

Brad Frost (2013): una metodología para estructurar la biblioteca de componentes.

Cinco niveles de composición

De lo más pequeño y reutilizable a la página completa.

Átomos

Botón, input, label, ícono

Moléculas

Search bar (input + button)

Organismos

Header completo, sidebar

Templates

Layout sin contenido real

Pages

Template + datos reales



La idea no es seguir la metáfora al pie de la letra — sino tener un vocabulario para hablar de composición entre diseño y código.

Beneficios

¿Por qué las grandes organizaciones invierten en esto?

Lo que ganamos

Eficiencia
Equipos replican diseños sin reinventarlos

Consistencia
Experiencia uniforme entre productos y canales

Lenguaje común
Diseñadores y devs hablan el mismo idioma

Escalabilidad
Crecer sin perder calidad ni identidad

Calidad
Componentes probados, accesibles y mantenidos

Onboarding
Las personas nuevas aprenden el sistema, no cada producto

Cuándo NO usar un Design System

No siempre el costo se justifica. Tres casos donde un Design System estorba más que ayuda.

Proyectos one-off

Una landing page de campaña que vivirá 2 semanas. La inversión nunca se amortiza.

Proyectos artísticos / experimentales

Cuando la innovación visual y la singularidad importan más que la consistencia.

Equipos sin capacidad de mantener

Un Design System que nadie mantiene se vuelve basura técnica rápidamente.

Un Design System es un compromiso de largo plazo. Sin equipo dedicado y sponsor ejecutivo, mejor empezar con un Style Guide simple.

¿Construir, adaptar o adoptar?

Tres caminos para tener un Design System en tu producto.

Tres estrategias

Adopt — adoptar uno existente

Costo: bajo

Control: mínimo

Ej: usar Material UI tal cual

Bueno para MVPs, productos internos, prototipos serios.

Adapt — adaptar uno existente

Costo: medio

Control: moderado

Ej: Tailwind + shadcn/ui personalizado

Equilibrio razonable para la mayoría de productos.

Create — crear el propio

Costo: alto

Control: total

Ej: IBM Carbon, Atlassian, Polaris

Solo si la marca o el negocio lo justifican.

Para la mayoría de los equipos: empieza adoptando, evoluciona a adaptar cuando duele, considera crear solo cuando ya escalaste.

Ejemplos del mundo real

Lo que se construye cuando el negocio puede pagarlo.

Sistemas notables

Los referentes que aparecen en cualquier conversación seria de Design Systems.

Material Design — Google

Pen-and-paper digital con elevación. Web, Android, iOS, Flutter. Material 3 introduce color dinámico (Material You).

Apple HIG

Human Interface Guidelines — el estándar de iOS / macOS / visionOS, fuerte en patrones de plataforma.

Microsoft Fluent

Lenguaje cross-platform: Windows, Office, Teams, Web.

IBM Carbon

Open-source, B2B-oriented, fuerte énfasis en accesibilidad y data viz.

Atlassian Design System

Detrás de Jira, Confluence, Trello. Excelente documentación de patrones.

Shopify Polaris

Pensado para terceros que construyen sobre Shopify — voz y tono extremadamente cuidados.

Pitfalls comunes

Errores típicos que matan un Design System.

Construirlo antes de necesitarlo

Diseñar abstracciones para problemas que aún no existen. Empezar por extraer patrones reales y repetidos.

Sin equipo dedicado

Si nadie es responsable, la calidad se erosiona en meses.

Documentación desactualizada

El código avanza, los docs se quedan atrás. La fuente de verdad debe ser el código vivo.

Demasiada flexibilidad

20 props por componente terminan permitiendo cualquier cosa — y deja de ser un sistema.

Ignorar accesibilidad

Si los componentes base no son accesibles, todo el producto hereda la deuda.

Tokens primitivos en componentes

Usar `blue-500` directo en un botón. Cambiar la marca rompe el sistema. Pasar siempre por tokens semánticos.

Caso práctico: shadcn/ui

El estándar de facto en React moderno — y muy probablemente lo que vas a usar en tu proyecto.

¿Qué es shadcn/ui?

Una colección de componentes que se copian al proyecto, no se instalan como librería.

No es una librería tradicional
No hay un import desde node_modules — el código vive en tu repo

Construido sobre Radix UI
Primitivos accesibles, sin estilos: focus management, keyboard nav, ARIA

Estilizado con Tailwind CSS
Tokens vía CSS variables, dark mode con una clase

CLI para añadir componentes
npx shadcn@latest add button → te copia el archivo

Tú eres dueño del código
Lo modificas como cualquier otro componente del proyecto

```
# 1. Inicializar el proyecto
$ npx shadcn@latest init

# 2. Añadir un componente
$ npx shadcn@latest add button
✓ Created components/ui/button.tsx
```

```
// 3. Usarlo en cualquier vista
import { Button } from "@/components/ui/button"

<Button variant="outline">Click me</Button>
```

Filosofía: Open Code. Tres caminos (adopt / adapt / create) se colapsan en uno: adoptas la base, pero adaptas todo lo que necesites desde el día uno.

Las piezas que usa shadcn

Cuatro tecnologías combinadas — cada una resuelve una capa distinta del Design System.

Radix UI

Primitivos sin estilos: `Dialog`, `Popover`, `Select`, `Tooltip` …

Focus management, keyboard nav y ARIA resueltos.

Tailwind CSS

Utility classes para estilo. Tokens vía CSS variables (`--primary`, `--muted` …).

Dark mode con `.dark` y variables redefinidas.

CVA (Class Variance Authority)

Define variantes de componente con tipado: `variant: 'default' | 'outline' | 'ghost'`.

Sustituye props con clases por una API tipada.

Lucide Icons

Set de íconos por defecto, intercambiable.

~1500 íconos, tree-shakeable, consistentes.

Anatomía de un componente shadcn

El `Button` que llega a tu proyecto cuando ejecutas `npx shadcn add button`.

```tsx
// components/ui/button.tsx
import { cva } from "class-variance-authority"

const buttonVariants = cva(
  "inline-flex items-center justify-center rounded-md tex
  {
    variants: {
      variant: {
        default: "bg-primary text-primary-foreground",
        outline: "border border-input bg-background",
        ghost: "hover:bg-accent",
      },
      size: {
        default: "h-10 px-4 py-2",
        sm: "h-9 px-3",
        lg: "h-11 px-8",
      },
    },
    defaultVariants: { variant: "default", size: "default
  }
)
```

Tokens semánticos

`bg-primary`, `text-primary-foreground` — no hex hardcoded.

Variantes tipadas

TypeScript autocompleta `variant` y `size`.

Editable

¿Necesitas un `variant: "danger"`? Lo agregas en este mismo archivo.

¿Preguntas?

Clase 14 - Design Systems: Teoría



PONTIFICIA
UNIVERSIDAD
CATÓLICA
DE CHILE

IIC2182 - Primer Semestre 2026