Ejemplos Prácticos de Lazy Loading

1. Lazy Loading de Formularios Complejos

Antes (sin lazy loading)

```
import { ContractForm } from './components/contract-form';

export default function ContractPage() {
  return (
    <div>
      <h1>Nuevo Contrato</h1>
      <ContractForm onSubmit={handleSubmit} />
    </div>
  );
}
```

Después (con lazy loading)

```
import { createLazyComponent } from '@/lib/lazy-components';
import { LoadingState } from '@/components/ui/loading-state';

const LazyContractForm = createLazyComponent(
  () => import('./components/contract-form').then(mod => ({ default:
mod.ContractForm })),
  'Cargando formulario de contrato...'
);

export default function ContractPage() {
  return (
    <div>
      <h1>Nuevo Contrato</h1>
      <LazyContractForm onSubmit={handleSubmit} />
    </div>
  );
}
```

Beneficio: El formulario (que puede pesar 50-100KB) solo se carga cuando se visita esta página.

---

2. Lazy Loading de Modales/Diálogos

Antes

```
import { DeleteConfirmDialog } from '@/components/ui/confirm-dialog';

export default function BuildingsPage() {
  const [showDelete, setShowDelete] = useState(false);

  return (
    <div>
      {/* ... lista de edificios ... */}
      <DeleteConfirmDialog
        open={showDelete}
        onConfirm={handleDelete}
      />
    </div>
  );
}
```

Después

```
import dynamic from 'next/dynamic';
import { LoadingSpinner } from '@/components/ui/loading-state';

const DeleteConfirmDialog = dynamic(
  () => import('@/components/ui/confirm-dialog').then(mod => ({ default: mod.ConfirmDi
alog })),
  {
    loading: () => <LoadingSpinner />,
    ssr: false // Los modales no necesitan SSR
  }
);

export default function BuildingsPage() {
  const [showDelete, setShowDelete] = useState(false);

  return (
    <div>
      {/* ... lista de edificios ... */}
      {showDelete && (
        <DeleteConfirmDialog
          open={showDelete}
          onConfirm={handleDelete}
        />
      )}
    </div>
  );
}
```

Beneficio: El diálogo solo se carga cuando el usuario hace click en “Eliminar”.

---

3. Lazy Loading de Tabs

Antes

```
import { Tabs, TabsContent, TabsList, TabsTrigger } from '@/components/ui/tabs';
import { GeneralTab } from './tabs/general-tab';
import { ModulesTab } from './tabs/modules-tab';
import { BrandingTab } from './tabs/branding-tab';

export default function CompanyDetailPage() {
  return (
    <Tabs defaultValue="general">
      <TabsList>
        <TabsTrigger value="general">General</TabsTrigger>
        <TabsTrigger value="modules">Módulos</TabsTrigger>
        <TabsTrigger value="branding">Branding</TabsTrigger>
      </TabsList>

      <TabsContent value="general">
        <GeneralTab />
      </TabsContent>
      <TabsContent value="modules">
        <ModulesTab />
      </TabsContent>
      <TabsContent value="branding">
        <BrandingTab />
      </TabsContent>
    </Tabs>
  );
}
```

Después

```tsx
import dynamic from 'next/dynamic';
import { Tabs, TabsContent, TabsList, TabsTrigger } from '@/components/ui/tabs';
import { LoadingState } from '@/components/ui/loading-state';

// Solo cargar el tab cuando se hace click
const GeneralTab = dynamic(() => import('./tabs/general-tab'), {
  loading: () => <LoadingState message="Cargando información general..." />,
  ssr: false
});

const ModulesTab = dynamic(() => import('./tabs/modules-tab'), {
  loading: () => <LoadingState message="Cargando módulos..." />,
  ssr: false
});

const BrandingTab = dynamic(() => import('./tabs/branding-tab'), {
  loading: () => <LoadingState message="Cargando personalización..." />,
  ssr: false
});

export default function CompanyDetailPage() {
  return (
    <Tabs defaultValue="general">
      <TabsList>
        <TabsTrigger value="general">General</TabsTrigger>
        <TabsTrigger value="modules">Módulos</TabsTrigger>
        <TabsTrigger value="branding">Branding</TabsTrigger>
      </TabsList>

      <TabsContent value="general">
        <GeneralTab />
      </TabsContent>
      <TabsContent value="modules">
        <ModulesTab />
      </TabsContent>
      <TabsContent value="branding">
        <BrandingTab />
      </TabsContent>
    </Tabs>
  );
}
```

Beneficio: Cada tab solo se carga cuando el usuario hace click, reduciendo el bundle inicial hasta un 70% si tienes 3 tabs.

---

4. Lazy Loading Condicional

Caso: Feature flag o permiso

```
import dynamic from 'next/dynamic';
import { usePermissions } from '@/lib/hooks/usePermissions';

const AdminPanel = dynamic(() => import('./components/admin-panel'), {
  ssr: false
});

export default function DashboardPage() {
  const { isAdmin } = usePermissions();

  return (
    <div>
      <h1>Dashboard</h1>
      {/* Solo carga el componente si el usuario es admin */}
      {isAdmin && <AdminPanel />}
    </div>
  );
}
```

Beneficio: Los usuarios normales nunca descargan el código del panel de admin.

---

5. Lazy Loading de Librerías Pesadas

Caso: Editor WYSIWYG

```
import dynamic from 'next/dynamic';
import 'react-quill/dist/quill.snow.css'; // CSS se carga siempre

const ReactQuill = dynamic(() => import('react-quill'), {
  ssr: false,
  loading: () => <div className="h-64 bg-gray-100 animate-pulse rounded" />
});

export default function ArticleEditor() {
  const [content, setContent] = useState('');

  return (
    <div>
      <h2>Editor de Artículo</h2>
      <ReactQuill value={content} onChange={setContent} />
    </div>
  );
}
```

Beneficio: React Quill pesa ~300KB. Solo se carga cuando se visita el editor.

---

6. Lazy Loading con Suspense (React 18+)

```
import { Suspense, lazy } from 'react';
import { LoadingState } from '@/components/ui/loading-state';

const HeavyChart = lazy(() => import('./components/heavy-chart'));

export default function AnalyticsPage() {
  return (
    <div>
      <h1>Analytics</h1>

      <Suspense fallback={<LoadingState message="Cargando gráficos..." />}>
        <HeavyChart data={data} />
      </Suspense>
    </div>
  );
}
```

---

7. Lazy Loading de Rutas Completas

En App Router de Next.js 13+

```
// app/admin/dashboard/loading.tsx
export default function Loading() {
  return <LoadingState message="Cargando dashboard..." />;
}

// app/admin/dashboard/page.tsx
// Este archivo se carga automáticamente de forma lazy
export default function AdminDashboard() {
  return <div>Dashboard content</div>;
}
```

---

8. Preloading Estratégico

```
import dynamic from 'next/dynamic';
import { useEffect } from 'react';

const HeavyComponent = dynamic(() => import('./heavy-component'), {
  ssr: false
});

export default function Page() {
  // Precargar después de 2 segundos (cuando el usuario ya vio el contenido inicial)
  useEffect(() => {
    const timer = setTimeout(() => {
      import('./heavy-component');
    }, 2000);
    return () => clearTimeout(timer);
  }, []);

  return (
    <div>
      <h1>Contenido principal</h1>
      {/* Se muestra después, pero ya estará precargado */}
      <HeavyComponent />
    </div>
  );
}
```

Checklist de Cuándo Usar Lazy Loading

✅ **SÍ usar lazy loading cuando**:
- El componente pesa más de 50KB
- No se muestra inmediatamente (tabs, modales)
- Es condicional (permisos, feature flags)
- Usa librerías pesadas (editores, charts complejos)
- Está en una ruta administrativa/poco visitada

❌ **NO usar lazy loading cuando**:
- Es contenido above-the-fold
- Es crítico para el funcionamiento (navegación, header)
- Es muy pequeño (<10KB)
- Se necesita para SEO
- Empeora la experiencia de usuario (loading constante)

Medición de Impacto

```
# Antes de lazy loading
yarn build
# Nota: Tamaño del chunk principal

# Después de lazy loading
yarn build
# Comparar tamaño y número de chunks

# Ver chunks generados
ls -lh .next/static/chunks/

# Bundle analyzer
ANALYZE=true yarn build
```

Errores Comunes y Soluciones

Error: “Cannot read property of undefined”

Causa: El componente exporta un named export, no default

```
// ❌ Incorrecto
const Comp = dynamic(() => import('./component'));

// ✅ Correcto
const Comp = dynamic(() => import('./component').then(mod => ({ default: mod.MyCompone
nt })));
```

Error: “Hydration mismatch”

Causa: Intentar hacer SSR de un componente que necesita el DOM

```
// ✅ Solución: Desactivar SSR
const Comp = dynamic(() => import('./component'), { ssr: false });
```

Performance: Loading state demasiado lento

Causa: El componente tarda en cargar

```
// ✅ Solución: Preload con intersección
import { useInView } from 'react-intersection-observer';

function LazySection() {
  const { ref, inView } = useInView({ triggerOnce: true });

  return (
    <div ref={ref}>
      {inView && <HeavyComponent />}
    </div>
  );
}
```