Guía de Migración - Optimización de Bundle

Migración Gradual: Paso a Paso

Esta guía te ayudará a aplicar las optimizaciones de bundle de forma segura y gradual.

Fase 1: Preparación (Sin cambios en producción)

1.1 Analizar estado actual

```
# 1. Build actual para baseline
cd /home/ubuntu/homming_vidaro/nextjs_space
yarn build

# 2. Analizar imports problemáticos
node scripts/analyze-imports.js > analysis-report.txt

# 3. Verificar tamaños actuales
node scripts/check-bundle-size.js
```

1.2 Revisar la documentación

```
# Leer la documentación completa
cat OPTIMIZACION_BUNDLE.md

# Revisar la configuración recomendada
cat next.config.optimized.js
```

1.3 Hacer backup

```
# Backup del next.config.js actual
cp next.config.js next.config.backup.js

# Tag en git (si usas git)
git tag -a v1.0-pre-optimization -m "Before bundle optimization"
```

Fase 2: Implementación de Lazy Loading (Bajo riesgo)

2.1 Verificar que existe el componente lazy

```
ls -la components/ui/lazy-charts-extended.tsx
```

✅ Si existe, continúa. Si no, créalo primero.

2.2 Migrar páginas con charts (una a la vez)

Lista de páginas a migrar:

- `/app/dashboard/page.tsx`
- `/app/analytics/page.tsx`
- `/app/bi/page.tsx`
- `/app/reportes/page.tsx`
- `/app/admin/dashboard/page.tsx`
- Otras páginas con gráficos

Proceso para cada página:

```
# 1. Hacer backup de la página
cp app/dashboard/page.tsx app/dashboard/page.tsx.backup

# 2. Buscar imports de recharts
grep -n "from 'recharts'" app/dashboard/page.tsx

# 3. Editar el archivo y cambiar imports
# ANTES:
# import { LineChart, XAxis, YAxis, ... } from 'recharts';

# DESPUÉS:
# import { LineChart, XAxis, YAxis, ... } from '@/components/ui/lazy-charts-extended';

# 4. Test en desarrollo
yarn dev
# Visitar http://localhost:3000/dashboard
# Verificar que los gráficos cargan correctamente

# 5. Si funciona, continuar con la siguiente página
# Si no funciona, restaurar backup:
# cp app/dashboard/page.tsx.backup app/dashboard/page.tsx
```

2.3 Buscar todos los archivos con recharts

```
# Encontrar todos los archivos que usan recharts
find app -type f -name "*.tsx" -exec grep -l "from 'recharts'" {} \;

# Guardar la lista
find app -type f -name "*.tsx" -exec grep -l "from 'recharts'" {} \; > recharts-
files.txt

# Migrar cada archivo según el proceso anterior
```

2.4 Verificar después de migración

```
# Build y verificar que no hay errores
yarn build

# Comparar tamaños
node scripts/check-bundle-size.js

# Debería ver reducción en el bundle principal
```

Fase 3: Aplicar next.config.js optimizado (Riesgo medio)

3.1 Revisar diferencias

```
# Comparar configuraciones
diff next.config.js next.config.optimized.js
```

3.2 Aplicar configuración

Opción A: Reemplazo completo (recomendado si el diff es simple)

```
# Backup
cp next.config.js next.config.old.js

# Aplicar nueva config
cp next.config.optimized.js next.config.js
```

Opción B: Merge manual (si tienes customizaciones)

1. Abrir ambos archivos lado a lado
2. Copiar secciones específicas de `next.config.optimized.js`
3. Mantener tus customizaciones existentes

3.3 Test intensivo en desarrollo

```
# Limpiar build anterior
rm -rf .next

# Build nuevo
yarn build

# Si hay errores, revisar los logs cuidadosamente
# Errores comunes:
# - null-loader no instalado -> yarn add -D null-loader
# - Sintaxis incorrecta en webpack config
```

3.4 Test de todas las páginas principales

```
# Iniciar dev server
yarn dev

# Visitar y verificar:
# - Homepage: http://localhost:3000
# - Dashboard: http://localhost:3000/dashboard
# - Analytics: http://localhost:3000/analytics
# - BI: http://localhost:3000/bi
# - Admin: http://localhost:3000/admin
# - Otras páginas críticas

# Verificar en consola del browser:
# - No hay errores de JavaScript
# - Charts cargan correctamente (puede tomar 1-2s con lazy loading)
# - No hay warnings de hidratación
```

Fase 4: Optimización de Imports (Bajo riesgo, alto impacto)

4.1 Ejecutar análisis

```
node scripts/analyze-imports.js > import-issues.txt
cat import-issues.txt
```

4.2 Priorizar correcciones

Alta prioridad (hacer primero):
- Wildcard imports de lucide-react
- Imports directos de recharts (deberían estar migrados ya)
- Imports completos de lodash

Media prioridad:
- Imports de date-fns sin ruta específica

Baja prioridad:
- Optimizaciones menores

4.3 Corregir imports (archivo por archivo)

Ejemplo con lucide-react:

```
# Encontrar archivos problemáticos
grep -r "import \* as.*from 'lucide-react'" app/

# Para cada archivo encontrado:
# ANTES:
# import * as Icons from 'lucide-react';
# <Icons.Home />

# DESPUÉS:
# import { Home, User, Settings } from 'lucide-react';
# <Home />
```

Ejemplo con lodash:

```
# ANTES:
# import _ from 'lodash';
# const mapped = _.map(array, fn);

# DESPUÉS:
# import { map } from 'lodash';
# const mapped = map(array, fn);
```

4.4 Verificar después de cada cambio

```
# Test rápido
yarn dev

# Build para verificar tamaño
yarn build
node scripts/check-bundle-size.js
```

Fase 5: Validación Final

5.1 Build completo y análisis

```
# Limpiar todo
rm -rf .next
rm -rf node_modules/.cache

# Build fresco
yarn build

# Analizar bundle final
node scripts/check-bundle-size.js

# Comparar con baseline inicial
# Deberías ver:
# - 40-50% reducción en bundle total
# - First Load JS < 800KB
# - No chunks > 1.5MB
```

5.2 Test de producción local

```
# Iniciar en modo producción
yarn build && yarn start

# Visitar http://localhost:3000
# Test completo de funcionalidad:
# - Login/Logout
# - Navegación entre páginas
# - Carga de gráficos
# - Formularios
# - Modales y componentes interactivos
```

5.3 Lighthouse audit

```
# Con Chrome DevTools:
# 1. F12 -> Lighthouse tab
# 2. Generar reporte
# 3. Verificar mejoras en Performance score

# Objetivos:
# - Performance: > 85
# - First Contentful Paint: < 2.0s
# - Largest Contentful Paint: < 2.5s
# - Total Blocking Time: < 300ms
```

Fase 6: Deployment

6.1 Pre-deployment checklist

- [ ] Todos los tests pasan
- [ ] Build de producción exitoso
- [ ] Bundle size dentro de límites
- [ ] No hay console.errors en browser
- [ ] Test manual de funcionalidad crítica OK
- [ ] Lighthouse score mejorado
- [ ] Backup de código anterior disponible

6.2 Deploy a staging (si disponible)

```
# Deploy a entorno de staging primero
# Verificar en staging antes de producción
```

6.3 Deploy a producción

```
# Según tu proceso de deploy
# Ejemplo con Vercel:
# vercel --prod

# O según tu pipeline de CI/CD
```

6.4 Post-deployment monitoring

Primeras 24 horas:
- Monitorear logs de errores
- Revisar métricas de performance
- Verificar que no hay incremento en errores 404/500
- User feedback

Primera semana:
- Analizar Core Web Vitals en producción
- Comparar métricas con baseline
- Recolectar feedback de usuarios

Rollback (Si algo sale mal)

Rollback de next.config.js

```
# Restaurar configuración anterior
cp next.config.old.js next.config.js

# Rebuild
yarn build

# Redeploy
```

Rollback de código

```
# Si usas git
git checkout v1.0-pre-optimization

# Rebuild y deploy
yarn build
# ... deploy process
```

Rollback parcial

Si solo algunas páginas tienen problemas:

```
# Restaurar archivos específicos
cp app/dashboard/page.tsx.backup app/dashboard/page.tsx

# Rebuild solo lo necesario
yarn build
```

Troubleshooting

Error: “null-loader” not found

```
yarn add -D null-loader
```

Error: Charts no se ven

Síntomas: Pantalla en blanco donde deberían estar los gráficos

Solución:
1. Verificar que el import es correcto:
`typescript`
`  import { LineChart } from '@/components/ui/lazy-charts-extended';`
2. Verificar que el componente lazy existe:
`bash`
`  ls -la components/ui/lazy-charts-extended.tsx`
3. Verificar en browser console por errores específicos

Error: Build timeout o out of memory

Solución:

```
# Incrementar memoria para build
NODE_OPTIONS="--max-old-space-size=8192" yarn build
```

Warning: “asset size limit”

No es un error, solo un warning. Pero indica que debes continuar optimizando.

Métricas de Éxito

Antes vs Después

Registra estas métricas antes y después:

```
| Métrica | Antes | Después | Mejora |
|---------|-------|---------|--------|
| Bundle Total | ____ MB | ____ MB | ___% |
| First Load JS | ____ KB | ____ KB | ___% |
| Build Time | ____ min | ____ min | ___% |
| FCP | ____ s | ____ s | ___% |
| LCP | ____ s | ____ s | ___% |
| Lighthouse Score | ____ | ____ | ___% |
```

Siguientes Pasos

Después de completar esta migración:

1. **Monitoreo continuo**
   - Agregar check de bundle size a CI/CD
   - Lighthouse CI en cada PR

2. **Optimizaciones adicionales**
   - Server Components (Next.js 13+)
   - Image optimization con next/image
   - Font optimization con next/font

3. **Educación del equipo**
   - Compartir OPTIMIZACION_BUNDLE.md con el equipo
   - Code review checklist para imports
   - Guidelines en documentation

Soporte

Si encuentras problemas durante la migración:

1. Revisa los logs detalladamente

2. Consulta OPTIMIZACION_BUNDLE.md
3. Ejecuta scripts de análisis:
   - `node scripts/analyze-imports.js`
   - `node scripts/check-bundle-size.js`
4. Revisa este documento de troubleshooting

¡Buena suerte con la migración! 🚀