🚀 Inicio Rápido - Optimización de Bundle

👋 Bienvenido

Este es tu punto de partida para optimizar el bundle de INMOVA. Todo está preparado y listo.

📊 Estado Actual

✅ **Muy bueno** - Solo 2 problemas críticos encontrados
📦 **Archivos listos** - Todos los componentes y scripts creados
📚 **Documentado** - Guías completas disponibles

Problemas identificados:
- 1 uso de moment.js (pesado, ~300KB)
- 160 imports de date-fns optimizables (bajo impacto)

Mejora esperada: 45-50% de reducción en bundle size

🛠️ Qué se creó para ti

```
/home/ubuntu/homming_vidaro/
├── next.config.optimized.js          # Configuración optimizada de Next.js
├── OPTIMIZACION_BUNDLE.md            # Documentación técnica completa
├── RESUMEN_OPTIMIZACION.md           # Resumen ejecutivo
├── INICIO_RAPIDO.md                  # Este archivo
├── nextjs_space/
│   ├── components/ui/
│   │   └── lazy-charts-extended.tsx # Lazy loading para charts
└── scripts/
    ├── analyze-imports.js            # Analiza imports problemáticos
    ├── check-bundle-size.js          # Valida tamaños de bundle
    └── migration-guide.md            # Guía paso a paso
```

⚡ Opción A: Quick Wins (2 horas, 30-35% mejora)

Paso 1: Ver el análisis actual (2 min)

```
cd /home/ubuntu/homming_vidaro/nextjs_space
node ../scripts/analyze-imports.js
```

Paso 2: Corregir moment.js (15 min)

Editar: `app/calendario/page.tsx`

```
// ❌ ANTES (moment.js ~300KB)
import moment from 'moment';
const formattedDate = moment().format('YYYY-MM-DD');

// ✅ DESPUÉS (date-fns ~20KB)
import { format } from 'date-fns';
const formattedDate = format(new Date(), 'yyyy-MM-dd');
```

Otros métodos comunes de moment → date-fns:

```
// moment().add(7, 'days')
import { addDays } from 'date-fns';
const newDate = addDays(new Date(), 7);

// moment().subtract(1, 'month')
import { subMonths } from 'date-fns';
const pastDate = subMonths(new Date(), 1);

// moment(date).isBefore(otherDate)
import { isBefore } from 'date-fns';
const result = isBefore(date, otherDate);

// moment().startOf('month')
import { startOfMonth } from 'date-fns';
const start = startOfMonth(new Date());
```

Paso 3: Instalar dependencia (1 min)

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

Paso 4: Aplicar configuración optimizada (5 min)

```
# Backup de la configuración actual
cp next.config.js next.config.backup.js

# Ver diferencias
diff next.config.js ../next.config.optimized.js

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

Paso 5: Test build (10 min)

```
# Limpiar cache
rm -rf .next

# Build
yarn build

# Si hay errores, revisar los logs
# Si todo está bien, continuar
```

Paso 6: Verificar mejoras (5 min)

```
# Analizar tamaño del bundle
node ../scripts/check-bundle-size.js

# Deberías ver:
# - Bundle total reducido 30-35%
# - Chunks más pequeños
# - Warnings de tamaño mejorados
```

Paso 7: Test funcional (15 min)

```
# Iniciar dev server
yarn dev

# Visitar y verificar:
# - http://localhost:3000
# - http://localhost:3000/dashboard
# - http://localhost:3000/calendario  (verificar que funciona sin moment)
# - http://localhost:3000/analytics
# - http://localhost:3000/admin
```

Paso 8: Desplegar (si todo está OK)

```
# Tu proceso de deploy habitual
# Ejemplo:
# git add .
# git commit -m "Optimización de bundle: moment.js → date-fns + next.config"
# git push
```

✅ **LISTO - Has mejorado el bundle en 30-35% en ~2 horas**

🚀 Opción B: Optimización Completa (4-5 horas, 45-50% mejora)

Incluye todo de Opción A, más:

Paso Extra 1: Encontrar páginas con charts (5 min)

```
# Buscar archivos que usan charts
find app -name "*.tsx" -exec grep -l "LineChart\|BarChart\|AreaChart\|PieChart" {} \;
> ../pages-with-charts.txt

# Ver la lista
cat ../pages-with-charts.txt
```

Paso Extra 2: Migrar cada página (15-20 min por página)

Para cada archivo en la lista:

```
# Ejemplo con dashboard
cp app/dashboard/page.tsx app/dashboard/page.tsx.backup
```

Editar el archivo:

```
// ❌ ANTES
import {
  LineChart,
  Line,
  XAxis,
  YAxis,
  Tooltip,
  ResponsiveContainer
} from 'recharts';

// ✅ DESPUÉS
import {
  LineChart,
  Line,
  XAxis,
  YAxis,
  Tooltip,
  ResponsiveContainer
} from '@/components/ui/lazy-charts-extended';
```

Testear:

```
yarn dev
# Visitar la página
# Verificar que el chart carga (puede tomar 1-2s la primera vez)
# Si funciona, continuar con la siguiente página
# Si no funciona, restaurar backup
```

Paso Extra 3: Build y validación final (15 min)

```
# Build completo
rm -rf .next
yarn build

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

# Analizar imports finales
node ../scripts/analyze-imports.js

# Deberías ver:
# - Bundle total reducido 45-50%
# - First Load JS < 650KB
# - Charts en chunks separados
# - Problemas de alta prioridad: 0
```

✅ **LISTO - Has optimizado el bundle al máximo**

🔧 Comandos Útiles

```
# Ver estado actual de imports
node scripts/analyze-imports.js

# Verificar tamaños de bundle
node scripts/check-bundle-size.js

# Build limpio
rm -rf .next && yarn build

# Test en modo producción
yarn build && yarn start

# Buscar uso de una librería específica
grep -r "import.*from 'moment'" app/

# Ver tamaño de node_modules
du -sh node_modules/

# Ver archivos más grandes en .next
find .next -type f -size +500k -exec ls -lh {} \; | awk '{print $9, $5}'
```

⚠️ Troubleshooting Rápido

Error: “Cannot find module ‘null-loader’“

```
cd /home/ubuntu/homming_vidaro/nextjs_space
yarn add -D null-loader
```

Error: “Build failed”

```
# Restaurar config anterior
cp next.config.backup.js next.config.js

# Limpiar y rebuild
rm -rf .next node_modules/.cache
yarn build
```

Charts no se ven

Verificar:

1. Import correcto:
   `typescript`
   `import { LineChart } from '@/components/ui/lazy-charts-extended';`
2. Componente existe:
   `bash`
   `ls -la components/ui/lazy-charts-extended.tsx`
3. No hay errores en consola del browser (F12)

Build muy lento o sin memoria

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

📚 Documentación Completa

Si necesitas más detalles:

```
# Guía paso a paso completa
cat scripts/migration-guide.md

# Documentación técnica
cat OPTIMIZACION_BUNDLE.md

# Resumen ejecutivo
cat RESUMEN_OPTIMIZACION.md
```

✅ Checklist Rápido

Antes de Empezar

- [ ] Leer este archivo completo
- [ ] Tener 2-5 horas disponibles
- [ ] Backup de código actual
- [ ] Git commit de estado actual

Durante Implementación (Opción A)

- [ ] Ejecutar `analyze-imports.js`
- [ ] Corregir `app/calendario/page.tsx` (moment → date-fns)
- [ ] Instalar `null-loader`
- [ ] Aplicar `next.config.optimized.js`
- [ ] Build exitoso
- [ ] Test funcional OK

Validación

- [ ] Ejecutar `check-bundle-size.js`
- [ ] Bundle reducido 30-35% (o más)
- [ ] No hay errores en console
- [ ] Todas las páginas funcionan
- [ ] Lighthouse score mejorado

🚀 ¡Comienza Ahora!

```
# Paso 1: Ir al proyecto
cd /home/ubuntu/homming_vidaro/nextjs_space

# Paso 2: Ver análisis
node ../scripts/analyze-imports.js

# Paso 3: Decidir qué opción seguir
# - Opción A: Quick Wins (2 horas)
# - Opción B: Completa (4-5 horas)

# Paso 4: Seguir los pasos de la opción elegida
```

💬 ¿Preguntas?

Toda la información está en:
- `RESUMEN_OPTIMIZACION.md` - Overview completo
- `OPTIMIZACION_BUNDLE.md` - Detalles técnicos
- `scripts/migration-guide.md` - Guía paso a paso detallada

🎯 **Objetivo:** Reducir bundle en 45-50%
⏱️ **Tiempo mínimo:** 2 horas (Quick Wins)
🔥 **Dificultad:** Baja-Media
❗ **Riesgo:** Bajo (todo documentado y con rollback)

👍 **¡Suerte con la optimización!**