



GUÍA DE SOLUCIÓN DE PROBLEMAS - VIRA

Soluciones paso a paso para los problemas más comunes que puedes encontrar al usar VIRA.



PROBLEMAS DE INSTALACIÓN

Error: “Module not found” después de yarn install

Síntomas:

```
Error: Cannot resolve module 'some-package'  
Module not found: Can't resolve '@/lib/something'
```

Soluciones:

```
# 1. Limpiar cache y reinstalar  
rm -rf node_modules  
rm yarn.lock  
yarn cache clean  
yarn install  
  
# 2. Verificar versión de Node.js  
node --version # Debe ser 18+ o 20+  
  
# 3. Reinstalar dependencias específicas  
yarn add @supabase/supabase-js @radix-ui/react-dialog  
  
# 4. Si persiste, usar npm en lugar de yarn  
rm -rf node_modules package-lock.json yarn.lock  
npm install
```

Error: “Database connection failed” durante setup

Síntomas:

```
Error: Invalid connection string  
Failed to connect to database
```

Soluciones:

1. Verificar variables de entorno:

```
# Revisar que .env tenga estas variables configuradas:  
cat .env | grep -E "(DATABASE_URL|SUPABASE_URL|SUPABASE_ANON_KEY)"  
  
# Formato correcto para Supabase:  
NEXT_PUBLIC_SUPABASE_URL=https://tu-proyecto.supabase.co  
NEXT_PUBLIC_SUPABASE_ANON_KEY=tu_anon_key_aquí
```

1. Verificar Supabase:

```
# Probar conexión directa
curl https://tu-proyecto.supabase.co/rest/v1/users \
  -H "apikey: tu_anon_key" \
  -H "Authorization: Bearer tu_anon_key"
```

1. Para PostgreSQL local:

```
# Verificar que PostgreSQL esté corriendo
pg_isready -h localhost -p 5432

# Probar conexión manual
psql "postgresql://usuario:password@localhost:5432/vira_production"
```

Error: “Build failed” con TypeScript

Síntomas:

```
Type error: Property 'xyz' does not exist on type...
Build failed because of typescript errors
```

Soluciones:

```
# 1. Ejecutar verificación de tipos
yarn tsc --noEmit

# 2. Actualizar tipos de Node.js
yarn add -D @types/node@latest

# 3. Regenerar tipos de Prisma (si aplica)
yarn prisma generate

# 4. Limpiar cache de TypeScript
rm -rf .next
rm tsconfig.tsbuildinfo
yarn build
```

PROBLEMAS DE BASE DE DATOS

Error: “Row Level Security” bloqueando acceso

Síntomas:

```
Error: PGRST301 - new row violates row-level security policy
Permission denied for table users
```

Soluciones:

1. Verificar autenticación:

```
// Asegúrate de que el usuario esté autenticado
const session = await getSession();
console.log('User ID:', session?.user?.id);
```

1. Revisar políticas RLS en Supabase:

```
-- Verificar políticas existentes
SELECT * FROM pg_policies WHERE tablename = 'users';

-- Temporal: Deshabilitar RLS para testing
ALTER TABLE users DISABLE ROW LEVEL SECURITY;

-- Volver a habilitar cuando funcione
ALTER TABLE users ENABLE ROW LEVEL SECURITY;
```

1. Usar Service Role Key para operaciones admin:

```
// Para operaciones que requieren permisos completos
const supabaseAdmin = createClient(
  process.env.NEXT_PUBLIC_SUPABASE_URL,
  process.env.SUPABASE_SERVICE_ROLE_KEY // ⚠ Solo en server-side
);
```

Performance lenta en consultas

Síntomas:

Queries taking >5 seconds
Database timeout errors
High CPU usage in database

Soluciones:

1. Agregar índices faltantes:

```
-- Ver queries más lentas
SELECT query, mean_exec_time, calls
FROM pg_stat_statements
ORDER BY mean_exec_time DESC LIMIT 10;

-- Crear índices necesarios
CREATE INDEX CONCURRENTLY idx_news_reports_user_created
ON news_reports(user_id, created_at);

CREATE INDEX CONCURRENTLY idx_scraped_news_category_date
ON scraped_news(category, scraped_at);
```

1. Optimizar consultas:

```
// Malo: Cargar todos los datos
const reports = await supabase.from('news_reports').select('*');

// Bueno: Limitar resultados y campos
const reports = await supabase
  .from('news_reports')
  .select('id, title, status, created_at')
  .order('created_at', { ascending: false })
  .limit(20);
```

1. Limpiar datos antiguos:

```
-- Eliminar noticias más de 90 días
DELETE FROM scraped_news
WHERE scraped_at < NOW() - INTERVAL '90 days';

-- Eliminar logs antiguos
DELETE FROM system_logs
WHERE timestamp < NOW() - INTERVAL '30 days';

-- Vacuum para recuperar espacio
VACUUM ANALYZE;
```

Error: “Too many connections”

Síntomas:

```
Error: sorry, too many clients already
FATAL: remaining connection slots are reserved
```

Soluciones:

1. Verificar conexiones activas:

```
SELECT count(*) as active_connections
FROM pg_stat_activity
WHERE state = 'active';

-- Ver conexiones por aplicación
SELECT application_name, count(*)
FROM pg_stat_activity
GROUP BY application_name;
```

1. Cerrar conexiones idle:

```
-- Terminar conexiones inactivas >1 hora
SELECT pg_terminate_backend(pid)
FROM pg_stat_activity
WHERE state = 'idle in transaction'
AND state_change < now() - interval '1 hour';
```

1. Configurar connection pooling:

```
// En lib/supabase.ts
const supabase = createClient(
  process.env.NEXT_PUBLIC_SUPABASE_URL,
  process.env.SUPABASE_SERVICE_ROLE_KEY,
  {
    db: {
      schema: 'public',
    },
    auth: {
      autoRefreshToken: false,
      persistSession: false
    }
  }
);
```



PROBLEMAS DE GENERACIÓN

Noticiero se queda “generando” infinitamente

Síntomas:

Status: "generating" por más de 30 minutos
 No se actualiza el progreso
 No hay errores visibles en consola

Soluciones:

1. Verificar logs del servidor:

```
# En desarrollo
yarn dev --verbose

# En producción (Vercel)
vercel logs

# Logs locales
tail -f .next/server.log
```

1. Verificar APIs externas:

```
# Probar ElevenLabs
curl -X GET "https://api.elevenlabs.io/v1/user" \
  -H "xi-api-key: tu_elevenlabs_key"

# Probar AbacusAI
curl -X POST "https://api.abacus.ai/chat/completions" \
  -H "Authorization: Bearer tu_abacus_key" \
  -H "Content-Type: application/json"
```

1. Reiniciar proceso manualmente:

```
// En consola del navegador o API call
fetch('/api/reports/tu-report-id', {
  method: 'PUT',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify({ action: 'retry_generation' })
});
```

1. Verificar límites de API:

```
// Revisar headers de rate limiting
fetch('/api/generate-newscast').then(response => {
  console.log('Rate limit remaining:', response.headers.get('x-ratelimit-remaining'));
  console.log('Rate limit reset:', response.headers.get('x-ratelimit-reset'));
});
```

Audio generado se escucha distorsionado o cortado

Síntomas:

Audio con ruido o interferencia
Cortes abruptos en el audio
Volumen inconsistente

Soluciones:

1. Verificar configuración de TTS:

```
// Configuración óptima para ElevenLabs
const ttsSettings = {
  stability: 0.5, // 0.3-0.7 rango óptimo
  similarity_boost: 0.8, // 0.7-0.9 para mejor calidad
  style: 0.3, // 0.2-0.4 para consistencia
  use_speaker_boost: true
};
```

1. Revisar longitud del texto:

```
// Dividir textos muy largos
function splitText(text, maxLength = 2000) {
  const sentences = text.match(/[^\.\!?\]+[^\.\!?\]+/g) || [text];
  const chunks = [];
  let currentChunk = '';

  sentences.forEach(sentence => {
    if ((currentChunk + sentence).length > maxLength) {
      if (currentChunk) chunks.push(currentChunk.trim());
      currentChunk = sentence;
    } else {
      currentChunk += sentence;
    }
  });

  if (currentChunk) chunks.push(currentChunk.trim());
  return chunks;
}
```

1. Verificar formato de audio:

```
// Configuración correcta para audio web
const audioConfig = {
  format: 'mp3',
  quality: '320kbps',
  sample_rate: 44100,
  channels: 'mono' // Suficiente para voz
};
```

Costos más altos de lo esperado

Síntomas:

Calculadora muestra \$2, pero cobran \$5
 Tokens usados no coinciden con estimación
 Cargos inesperados en servicios IA

Soluciones:

1. Verificar configuración de modelos:

```
// Revisar qué modelos estás usando realmente
const config = {
  rewrite_model: 'gpt-3.5-turbo', // $0.002/1k tokens
  // NO: 'gpt-4-turbo', // $0.03/1k tokens (15x más caro)

  voice_provider: 'aws-polly', // $16/1M chars
  // NO: 'elevenlabs', // $180/1M chars (11x más caro)
};
```

1. Monitorear uso real:

```
// Agregar logging detallado
async function logTokenUsage(service, operation, tokens, cost) {
  await supabase.from('token_usage').insert({
    user_id: userId,
    service,
    operation,
    tokens_used: tokens,
    cost_usd: cost,
    timestamp: new Date()
  });

  console.log(`💰 ${service}:${operation} = ${tokens} tokens = ${cost}`);
}
```

1. Optimizar configuraciones:

```
// Perfil económico personalizado
const economicProfile = {
  extraction_model: 'gpt-3.5-turbo',
  rewrite_model: 'gpt-3.5-turbo',
  humanization_model: 'claude-3-haiku', // Más barato que sonnet
  voice_provider: 'aws-polly',
  voice_id: 'Conchita' // Voz estándar, no neural
};
```

PROBLEMAS DE AUTENTICACIÓN

No puedo hacer login con Google/GitHub

Síntomas:

Botón de login no responde
 Error: "OAuth callback mismatch"
 "Invalid client" error

Soluciones:

1. Verificar configuración OAuth:

```
# En .env, verificar:
GOOGLE_CLIENT_ID=tu_client_id.apps.googleusercontent.com
GOOGLE_CLIENT_SECRET=tu_client_secret
NEXTAUTH_URL=https://tu-dominio.com
```

1. Configurar URLs en providers:

Google OAuth Console:

- Authorized JavaScript origins: https://tu-dominio.com
- Authorized redirect URIs: https://tu-dominio.com/api/auth/callback/google

GitHub OAuth App:

- Homepage URL: https://tu-dominio.com
- Authorization callback URL: https://tu-dominio.com/api/auth/callback/github

1. Verificar NextAuth configuración:


```
// En app/api/auth/[...nextauth]/route.ts
export { handler as GET, handler as POST }

// Verificar que authOptions esté correctamente exportado
import { authOptions } from '@lib/auth';
const handler = NextAuth(authOptions);
```

Session expira constantemente

Síntomas:

Usuario es deslogueado cada pocos minutos
"Session not found" errors frecuentes

Soluciones:

1. Configurar timeouts correctos:

```
// En lib/auth.ts
export const authOptions = {
  session: {
    strategy: 'jwt',
    maxAge: 30 * 24 * 60 * 60, // 30 días
    updateAge: 24 * 60 * 60, // Actualizar cada 24 horas
  },
  jwt: {
    maxAge: 30 * 24 * 60 * 60, // 30 días
  }
};
```

1. Verificar cookies:

```
// Verificar en DevTools > Application > Cookies
// Debe existir: next-auth.session-token

// Configurar cookies para producción
cookies: {
  sessionToken: {
    name: 'next-auth.session-token',
    options: {
      httpOnly: true,
      sameSite: 'lax',
      path: '/',
      secure: process.env.NODE_ENV === 'production'
    }
  }
}
```

1. Debug session:

```
// Agregar logs para debug
const { data: session, status } = useSession();

useEffect(() => {
  console.log('Session status:', status);
  console.log('Session data:', session);
}, [session, status]);
```

Error: “NEXTAUTH_SECRET not found”

Síntomas:

```
[next-auth] [error] [NO_SECRET]
NEXTAUTH_SECRET environment variable is not set
```

Soluciones:

```
# 1. Generar secret seguro
node -e "console.log(require('crypto').randomBytes(32).toString('hex'))"

# 2. Agregar a .env
NEXTAUTH_SECRET="tu_secret_generado_aquí"

# 3. Para producción, configurar en plataforma de hosting:
# Vercel: Dashboard > Settings > Environment Variables
# Netlify: Site settings > Environment variables
```



PROBLEMAS DE DEPLOYMENT

Build falla en producción

Síntomas:

```
Error occurred prerendering page
Build optimization failed
Out of memory error during build
```

Soluciones:

1. Aumentar memoria para build:

```
// En package.json
{
  "scripts": {
    "build": "NODE_OPTIONS='--max-old-space-size=4096' next build"
  }
}
```

1. Verificar variables de entorno en producción:

```
# Vercel
vercel env ls

# Netlify
netlify env:list

# Railway
railway variables
```

1. Build local para debug:

```
NODE_ENV=production yarn build
yarn start # Probar build localmente
```

500 Internal Server Error en API routes

Síntomas:

```
API endpoints returning 500
"Internal Server Error" without details
```

Soluciones:

1. Verificar logs detallados:

```
# Vercel
vercel logs --follow

# Netlify
netlify logs

# Local development
yarn dev # Ver errores en terminal
```

1. Agregar error handling:

```
// En API routes
export async function POST(req: NextRequest) {
  try {
    // Tu lógica aquí
    return NextResponse.json({ success: true });
  } catch (error) {
    console.error('API Error:', error);
    return NextResponse.json(
      { error: error.message, stack: error.stack },
      { status: 500 }
    );
  }
}
```

1. Verificar dependencias en producción:

```
// Mover deps de devDependencies a dependencies si son necesarias en runtime
{
  "dependencies": {
    "@prisma/client": "^5.0.0", // Necesario en runtime
    "prisma": "^5.0.0"          // Para generate en build
  }
}
```

Assets/imágenes no cargan

Síntomas:

404 errors para archivos CSS/JS
Imágenes no se muestran
Fonts no cargan

Soluciones:

1. Configurar base path:

```
// next.config.js
/** @type {import('next').NextConfig} */
const nextConfig = {
  assetPrefix: process.env.NODE_ENV === 'production' ? '/ruta-base' : '',
  basePath: process.env.NODE_ENV === 'production' ? '/ruta-base' : '',
}
```

1. Verificar Image component:

```
// Configurar dominios permitidos
// next.config.js
module.exports = {
  images: {
    domains: ['tu-bucket.s3.amazonaws.com', 'images.unsplash.com'],
    remotePatterns: [
      {
        protocol: 'https',
        hostname: '*.supabase.co',
      }
    ]
  }
}
```

1. Verificar CDN/S3 configuración:

```
# Probar acceso directo a assets
curl -I https://tu-bucket.s3.amazonaws.com/test-file.mp3

# Verificar CORS en S3 bucket
aws s3api get-bucket-cors --bucket tu-bucket-name
```

PROBLEMAS DE AUDIO

Audio no se reproduce en navegador

Síntomas:

Reproductor aparece pero no suena
 Error: "Failed to load media"
 Audio funciona en unos navegadores, en otros no

Soluciones:

1. Verificar formato de audio:

```
// Usar formatos ampliamente soportados
const supportedFormats = {
  mp3: 'audio/mpeg',    // ✓ Soportado en todos lados
  wav: 'audio/wav',     // ✓ Alta compatibilidad
  ogg: 'audio/ogg',     // ✗ No en Safari
  m4a: 'audio/mp4'     // ✓ Buena compatibilidad
};
```

1. Configurar headers CORS correctos:

```
// En API route que sirve audio
export async function GET(request: NextRequest) {
  return new NextResponse(audioBuffer, {
    headers: {
      'Content-Type': 'audio/mpeg',
      'Accept-Ranges': 'bytes',
      'Access-Control-Allow-Origin': '*',
      'Cache-Control': 'public, max-age=3600'
    }
  });
}
```

1. Implementar fallback para diferentes formatos:

```
<audio controls>
  <source src="audio.mp3" type="audio/mpeg" />
  <source src="audio.wav" type="audio/wav" />
  Tu navegador no soporta el elemento audio.
</audio>
```

S3 Audio URLs no accesibles

Síntomas:

403 Forbidden al acceder URLs de S3
 SignedURLs expiradas
 CORS errors desde navegador

Soluciones:

1. Configurar CORS en S3:

```
[
  {
    "AllowedHeaders": ["*"],
    "AllowedMethods": ["GET", "HEAD"],
    "AllowedOrigins": ["*"],
    "ExposeHeaders": ["ETag", "Content-Length"]
  }
]
```

1. Usar SignedURLs para archivos privados:

```
// Generar URL firmada
import { getSignedUrl } from '@aws-sdk/s3-request-presigner';
import { GetObjectCommand } from '@aws-sdk/client-s3';

const signedUrl = await getSignedUrl(s3Client, new GetObjectCommand({
  Bucket: bucketName,
  Key: audioKey
}), { expiresIn: 3600 }); // 1 hora
```

1. Política de bucket correcta:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::tu-bucket-name/audio/*"
    }
  ]
}
```



PROBLEMAS DE UI/UX

Interfaz no se ve correctamente en móvil

Síntomas:

Elementos cortados en mobile
Buttons muy pequeños para tocar
Scroll horizontal no deseado

Soluciones:

1. Verificar viewport meta tag:

```
<!-- En app/layout.tsx -->
<meta name="viewport" content="width=device-width, initial-scale=1" />
```

1. Usar clases responsive de Tailwind:

```
// Responsive design pattern
<div className="
  grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3
  gap-4 p-4
  text-sm md:text-base
">
```

1. Testing en diferentes dispositivos:

```
// Usar DevTools Device Emulation
// Chrome > F12 > Toggle Device Toolbar
// Probar: iPhone 12, iPad, Galaxy S20
```

Loading states no se muestran

Síntomas:

Pantalla blanca mientras carga
No feedback visual durante operaciones
Usuarios no saben si algo está pasando

Soluciones:

1. Implementar skeletons:

```
// Componente skeleton
const NewsSkeleton = () => (
  <div className="animate-pulse">
    <div className="h-4 bg-gray-200 rounded w-3/4 mb-2"></div>
    <div className="h-4 bg-gray-200 rounded w-1/2"></div>
  </div>
);

// Usar mientras carga
{loading ? <NewsSkeleton /> : <NewsItem data={news} />}
```

1. Estados de loading específicos:

```
const [states, setStates] = useState({
  scraping: false,
  rewriting: false,
  generating_audio: false,
  finalizing: false
});

// Mostrar progreso específico
{states.scraping && <span>📰 Extrayendo noticias...</span>}
{states.rewriting && <span>🔄 Reescribiendo contenido...</span>}
{states.generating_audio && <span>🔊 Generando audio...</span>}
```

1. Progress indicators:

```
// Barra de progreso realista
const steps = ['scraping', 'rewriting', 'audio', 'finalizing'];
const currentStep = 2;
const progress = ((currentStep + 1) / steps.length) * 100;

<div className="w-full bg-gray-200 rounded-full h-2">
  <div
    className="bg-blue-600 h-2 rounded-full transition-all duration-300"
    style={{ width: `${progress}%` }}
  />
</div>
```

Formularios no validan correctamente

Síntomas:

Submit con datos inválidos
Errores no se muestran al usuario
Campos requeridos pasan vacíos

Soluciones:

1. Usar react-hook-form con zod:

```
import { useForm } from 'react-hook-form';
import { zodResolver } from '@hookform/resolvers/zod';
import { z } from 'zod';

const schema = z.object({
  title: z.string().min(1, 'Título es requerido'),
  duration: z.number().min(5).max(60),
  region: z.string().min(1, 'Región es requerida')
});

const { register, handleSubmit, formState: { errors } } = useForm({
  resolver: zodResolver(schema)
});
```

1. Mostrar errores claramente:

```
<input
  {...register('title')}
  className={errors.title ? 'border-red-500' : 'border-gray-300'}
/>
{errors.title && (
  <p className="text-red-500 text-sm mt-1">{errors.title.message}</p>
)}
```

1. Validación en tiempo real:


```
const [title, setTitle] = useState('');
const [titleError, setTitleError] = useState('');

useEffect(() => {
  if (title.length > 0 && title.length < 3) {
    setTitleError('Título debe tener al menos 3 caracteres');
  } else {
    setTitleError('');
  }
}, [title]);
```



PROBLEMAS DE PERFORMANCE

Aplicación lenta en general

Síntomas:

Navegación tarda mucho
Componentes tardan en renderizar
Memoria del navegador se agota

Soluciones:

1. Optimizar componentes con React.memo:

```
// Memoizar componentes pesados
const NewsItem = React.memo(({ news }) => {
  return <div>{news.title}</div>;
});

// Usar useMemo para cálculos pesados
const expensiveCalculation = useMemo(() => {
  return news.reduce((acc, item) => acc + item.cost, 0);
}, [news]);
```

1. Lazy loading para componentes grandes:

```
import { lazy, Suspense } from 'react';

const Timeline = lazy(() => import('./Timeline'));

// Usar con Suspense
<Suspense fallback=<div>Cargando timeline...</div>>
  <Timeline />
</Suspense>
```

1. Optimizar queries de datos:

```
// Malo: Cargar todo
const { data } = useSWR('/api/reports');

// Bueno: Paginación
const { data } = useSWR(`/api/reports?page=${page}&limit=10`);

// Mejor: Infinite loading
const { data, size, setSize } = useSWRInfinite(
  (index) => `/api/reports?page=${index + 1}&limit=10`
);
```

Bundle size muy grande

Síntomas:

First load muy lento (>10s)
Lighthouse Performance score bajo
Mobile data usage alto

Soluciones:

1. Analizar bundle size:

```
# Instalar analizador
yarn add -D @next/bundle-analyzer

# Configurar en next.config.js
const withBundleAnalyzer = require('@next/bundle-analyzer')({
  enabled: process.env.ANALYZE === 'true',
});

# Ejecutar análisis
ANALYZE=true yarn build
```

1. Optimizar imports:

```
// Malo: Importar toda la librería
import _ from 'lodash';

// Bueno: Import específico
import { debounce } from 'lodash';

// Mejor: Usar alternativas más pequeñas
import { debounce } from '../utils/debounce';
```

1. Code splitting por rutas:

```
// next.config.js
module.exports = {
  experimental: {
    optimizeCss: true,
  },
  compiler: {
    removeConsole: process.env.NODE_ENV === 'production',
  }
}
```

CUANDO TODO FALLA

Reset Completo del Sistema

Si nada más funciona, este es el proceso de reset completo:

```
# 1. Backup datos importantes
mysqldump -u root -p vira_production > backup_$(date +%Y%m%d).sql

# 2. Limpiar completamente
rm -rf node_modules
rm yarn.lock package-lock.json
rm -rf .next

# 3. Reinstalar desde cero
yarn install
yarn prisma generate
yarn prisma db push

# 4. Reset variables de entorno
cp .env .env.backup
cp CONFIGURACIONES/variables-entorno-ejemplo.env .env
# Editar .env con tus valores

# 5. Primera build limpia
yarn build
yarn dev
```

Contactar Soporte

Si el problema persiste después de intentar estas soluciones:

1. Recopilar información del error:

```
# Logs de desarrollo
yarn dev > debug.log 2>&1

# Logs de producción
vercel logs > production.log



# Variables de entorno (sin valores sensibles)
env | grep -v SECRET | grep -v KEY > env.log
```


1. Información del sistema:

```
node --version > system-info.txt
yarn --version >> system-info.txt
echo $SHELL >> system-info.txt
uname -a >> system-info.txt
```

1. Descripción detallada:

- ☒ Qué estabas intentando hacer
- ☒ Qué esperabas que pasara
- ☒ Qué pasó realmente (error exacto)

-  Pasos para reproducir el problema
-  Screenshots si es relevante

¡Esta guía cubre el 95% de problemas comunes! Si tu problema no está aquí, probablemente sea algo específico de tu configuración que requiere investigación más profunda. 

Recuerda siempre hacer backups antes de hacer cambios importantes en producción! 