Entwickler-Report: crawlix.io — Implementation Guide

crawlix.io

2026-03-31

Entwickler-Report: crawlix.io — Implementation Guide

Inhaltsverzeichnis

Entwickler-Report: crawlix.io – Implementation Guide

Datum: 2026-03-31 **Framework:** Next.js 15 (App Router) auf Vercel **Sprachen:** DE (Haupt), EN, CS **Audit-Score:** 57/100 (F)

Inhaltsverzeichnis

Score-Übersicht

KATEGORIE	SCORE	NOTE	PRIORITÄT
Technical SEO	78/100	C+	Woche 3-4
Content Quality	61/100	D+	Monat 2
On-Page SEO	52/100	F	Woche 2
Schema / Structured Data	42/100	F	Woche 2
Performance (CWV)	65/100	D	Woche 3-4
Images	70/100	C	Monat 2
AI Search Readiness	38/100	F	Woche 1
Backlink Profile	5/100	F	Extern
Accessibility	91/100	A	Woche 3-4
Gesamt	**57/100**	**F**	

Erwarteter Score nach allen Fixes: ~85/100 (B)

Phase 1: Critical Fixes (diese Woche)

> **Erwarteter Score-Sprung: 57 –> ~65/100**

1.1 Duplicate Content eliminieren (EN/CS Blog)

Problem: 30 von 45 Sitemap-URLs liefern identischen deutschen Content unter `/en/` und `/cs/` mit falschem `lang`-Attribut. Das ist der gravierendste Ranking-Killer.

Betroffene URLs: Alle 10 Blog-Posts x 2 Sprachen (EN + CS) = 20 URLs

Lösung A (empfohlen – schnellste Variante): noindex per Middleware

Erstelle oder erweitere `middleware.ts` im Projekt-Root:

```
// middleware.ts
import { NextResponse } from 'next/server';
import type { NextRequest } from 'next/server';

// Blog-Pfade die noch nicht übersetzt sind
const UNTRANSLATED_BLOG_LOCALES = ['en', 'cs'];

export function middleware(request: NextRequest) {
  const { pathname } = request.nextUrl;

  // EN/CS Blog-Posts auf noindex setzen bis Übersetzungen vorhanden
  const isBlogPost = /^\/(en|cs)\/blog\/.+/.test(pathname);

  if (isBlogPost) {
    const response = NextResponse.next();
    response.headers.set('X-Robots-Tag', 'noindex, nofollow');
    return response;
  }

  return NextResponse.next();
}

export const config = {
  matcher: ['/(en|cs)/blog/:path*'],
};
```

Lösung B (alternative Variante): Meta-Tag im Blog Layout

Falls du das lieber auf Komponenten-Ebene löst:

```tsx
// app/[locale]/blog/[slug]/layout.tsx
import { headers } from 'next/headers';

export async function generateMetadata({ params }: { params: { locale: string; slug: string } }) {
  const { locale } = params;

  // Blog-Posts nur auf DE indexierbar
  if (locale !== 'de') {
    return {
      robots: {
        index: false,
        follow: false,
      },
    };
  }

  // ... bestehende Metadata-Logik
}
```

Lösung C (Sitemap bereinigen): EN/CS Blog aus Sitemap entfernen

In der Sitemap-Generierung (vermutlich `app/sitemap.ts` oder Plugin):

```ts
// app/sitemap.ts
import { MetadataRoute } from 'next';

export default function sitemap(): MetadataRoute.Sitemap {
  const locales = ['de', 'en', 'cs'];
  const blogSlugs = [
    'chatgpt-seo-sichtbarkeit',
    'onpage-seo-checkliste',
    'was-kostet-seo-audit',
    // ... alle 10 Slugs
  ];

  const pages: MetadataRoute.Sitemap = [];

  // Hauptseiten: alle 3 Sprachen
  const mainPages = ['', '/seo-check', '/partner', '/preise', '/blog'];
  for (const locale of locales) {
    for (const page of mainPages) {
      pages.push({
        url: `https://crawlix.io/${locale}${page}`,
        lastModified: new Date(),
        alternates: {
          languages: {
            de: `https://crawlix.io/de${page}`,
            en: `https://crawlix.io/en${page}`,
            cs: `https://crawlix.io/cs${page}`,
            'x-default': `https://crawlix.io${page}`,
          },
        },
      });
    }
  }

  // Blog-Posts: NUR Deutsch (bis Übersetzungen vorhanden)
  for (const slug of blogSlugs) {
    pages.push({
      url: `https://crawlix.io/de/blog/${slug}`,
      lastModified: new Date(),
      // Kein hreflang für EN/CS solange nicht übersetzt
    });
  }

  return pages;
}
```

Aufwand: 30 Min **Impact:** Eliminiert 67% Duplicate Content, verbessert Crawl-Budget massiv

---

1.2 llms.txt erstellen und deployen

Problem: crawlix bewirbt llms.txt als Deliverable (“Fertiger Code inklusive: robots.txt, llms.txt, JSON-LD Schemas”), hat selbst keine. Glaubwürdigkeits-Killer.

Lösung: Datei in `public/llms.txt` anlegen:

```
# crawlix.io

> White-Label SEO-Audit-Service für Agenturen und Entwickler. 8-Module Audit mit GEO/AI-Readiness, fertigen Code-Snippets und priorisiertem Aktionsplan. Lieferzeit 48 Stunden.

## Angebot
- Starter (400 EUR): 6 Module, 30+ Seiten Report, White-Label
- Professional (600 EUR): 8 Module, 50+ Seiten, GEO-Check, Wettbewerber-Gap
- Complete (1.200 EUR): Alles + 11-Kapitel Playbook, Implementierungs-Support, 30 Tage Nachbetreuung

## Zielgruppe
Web-Agenturen und Entwickler die SEO-Audits unter eigenem Branding verkaufen wollen.

## Module
Technical SEO, GEO/AI-Search, Schema.org, Content & E-E-A-T, Image SEO, Sitemap & Crawl, Deep Page Analyse, Wettbewerber-Gap

## Kontakt
- Website: https://crawlix.io
- E-Mail: contact@crawlix.io
- Gründer: Lukas Lavicka

## Weitere Informationen
- Preise: https://crawlix.io/de/preise
- Partner-Programm: https://crawlix.io/de/partner
- Blog: https://crawlix.io/de/blog
- Kostenloser SEO-Check: https://crawlix.io/de/seo-check
```

Verifikation nach Deploy:

```
curl -I https://crawlix.io/llms.txt
# Erwartet: HTTP 200, Content-Type: text/plain
```

Aufwand: 15 Min **Impact:** AI Readiness llms.txt-Komponente: 0 –> 80

1.3 MwSt-Widerspruch fixen

Problem: Das Impressum sagt “Kleinunternehmerregelung gemäß § 19 UStG – die Umsatzsteuer wird nicht erhoben”. Die Preise-Seite sagt “Alle Preise sind Einkaufspreise für Partner (netto, zzgl. MwSt.)”. Beides gleichzeitig ist **rechtlich unmöglich**.

Betroffene Datei: Vermutlich `messages/de.json` oder die Preise-Seite direkt

Fix – Text auf der Preise-Seite ändern:

```
// VORHER (falsch):
"Alle Preise sind Einkaufspreise für Partner (netto, zzgl. MwSt.)"

// NACHHER (korrekt bei Kleinunternehmerregelung):
"Alle Preise sind Endpreise. Gemäß § 19 UStG wird keine Umsatzsteuer erhoben."
```

Zusätzlich den Hinweis “USt-ID-Handling nach deutschem Recht” bei der Rechnungsbeschreibung entfernen oder anpassen – als Kleinunternehmer wird keine USt auf Rechnungen ausgewiesen.

Aufwand: 10 Min **Impact:** Rechtliches Risiko eliminiert (Business Logic Score: 45 –> 70 in Preiskonsistenz)

1.4 i18n-Key-Leak auf Partner-Seite fixen

Problem: Die Partner-Seite zeigt Raw-Translation-Keys statt Content:

```
partner.packages.starter.f1
partner.packages.starter.f2
partner.packages.professional.tag
```

Diagnose: Die i18n-Lookup schlägt fehl. Die Keys existieren nicht in der Locale-Datei.

Fix: Fehlende Keys in der Übersetzungsdatei ergänzen. Vermutlich in `messages/de.json` (oder `locales/de.json`, je nach i18n-Setup):

```
{
  "partner": {
    "packages": {
      "starter": {
        "tag": "Basis",
        "f1": "6 Analyse-Module",
        "f2": "30+ Seiten Report",
        "f3": "Schema-Analyse",
        "f4": "White-Label Branding",
        "f5": "Email-Lieferung in 48h"
      },
      "professional": {
        "tag": "Beliebt",
        "f1": "8 Analyse-Module",
        "f2": "50+ Seiten Report",
        "f3": "GEO/AI-Readiness Check",
        "f4": "Wettbewerbs-Gap-Analyse",
        "f5": "4-Phasen Aktionsplan"
      },
      "complete": {
        "tag": "Premium",
        "f1": "Alles aus Professional",
        "f2": "11-Kapitel Playbook",
        "f3": "Implementierungs-Support",
        "f4": "30 Tage Nachbetreuung",
        "f5": "Priority 24h Lieferung"
      }
    }
  }
}
```

Verifikation: Nach Deploy `/de/partner` aufrufen und prüfen, ob die Features korrekt angezeigt werden.

Aufwand: 30 Min **Impact:** Rendering-Bug behoben, Partner-Seite vollständig nutzbar

---

1.5 Starter-Paket Seitenzahl-Widerspruch fixen

Problem: Die Homepage zeigt BEIDE Angaben für das Starter-Paket: - Pricing-Sektion: “15-seitiger Report” - Solution-Sektion: “30+ Seiten Report”

Die Preise-Seite und Partner-Seite sagen konsistent “30+ Seiten”.

Fix: In der Homepage-Pricing-Sektion “15-seitiger Report” auf “30+ Seiten Report” ändern. ODER – falls beides korrekt ist (15-Seiten Executive Summary als Teil eines 30+-Seiten-Reports) – die Formulierung klarstellen:

```
// VORHER (widersprüchlich):
"15-seitiger Report"

// NACHHER (Option A -- wenn 30+ korrekt ist):
"30+ Seiten Report"

// NACHHER (Option B -- wenn beides stimmt):
"30+ Seiten Report inkl. 15-seitiger Executive Summary"
```

Aufwand: 5 Min **Impact:** Preiskonsistenz hergestellt

1.6 “98% Genauigkeit” Claim entfernen oder belegen

Problem: Die Homepage und EN-Version zeigen “98% Genauigkeit” ohne Quelle, Methodik oder Definition. Das ist ein konkreter, messbarer Claim der nach UWG § 5 (irreführende Werbung) angreifbar ist.

Betroffene Stellen: - `/de` – “So funktioniert’s”-Sektion - `/en` – Statistiken-Bereich

Fix (schnellste Variante):

```
// VORHER:
"98% Genauigkeit"

// NACHHER:
"Agentur-Qualität" oder "Manuell validierte Checks"
```

Aufwand: 5 Min **Impact:** Rechtliches Risiko eliminiert

1.7 Partner-Seite: Unique Title & Meta Description

Problem: Title und Meta Description der Partner-Seite sind identisch mit der Homepage – Keyword-Kannibalisierung.

Aktuell:

```
Title: "SEO-Audit mit AI-Search Check | Crawlix.io"
Description: "SEO wie in der Medizin: Diagnose, Plan, Umsetzung..."
```

Fix in der Partner-Page Metadata:

```tsx
// app/[locale]/partner/page.tsx (oder layout.tsx)
export const metadata = {
  title: 'Partner-Programm | White-Label SEO-Audits | Crawlix.io',
  description: 'Verkaufe SEO-Audits unter deinem Branding. 160-260% Marge, kein SEO-Team nötig. White-Label Reports
       ab 400 EUR mit 48h Lieferzeit.',
  openGraph: {
    title: 'Partner-Programm | White-Label SEO-Audits | Crawlix.io',
    description: 'Verkaufe SEO-Audits unter deinem Branding. 160-260% Marge, kein SEO-Team nötig.',
    url: 'https://crawlix.io/de/partner',
  },
};
```

Aufwand: 15 Min **Impact:** Duplicate Meta eliminiert, eigenes Ranking-Potenzial für Partner-Seite

---

Phase 2: Schema & On-Page (Woche 2)

> **Erwarteter Score-Sprung: ~65 –> ~75/100**

2.1 Product/Offer Schema für Preise-Seite

Problem: 3 Pakete (Starter 400 EUR, Professional 600 EUR, Complete 1.200 EUR) ohne Structured Data. Verpasste Rich-Result-Chance.

Lösung: JSON-LD Block in die Preise-Seite einfügen.

Erstelle eine Schema-Komponente oder füge direkt in die Preise-Page ein:

```tsx
// components/seo/PricingSchema.tsx
export function PricingSchema() {
  const schema = {
    '@context': 'https://schema.org',
    '@type': 'WebApplication',
    name: 'Crawlix SEO-Audit',
    description:
      'White-Label SEO-Audit-Tool für Agenturen. 8-Modul-Report mit AI-Readiness-Check, technischem Audit und priorisiertem Aktionsplan.',
    url: 'https://crawlix.io/de/preise',
    applicationCategory: 'BusinessApplication',
    operatingSystem: 'Web',
    browserRequirements: 'Moderner Browser (Chrome, Firefox, Safari, Edge)',
    inLanguage: ['de', 'en', 'cs'],
    author: {
      '@type': 'Organization',
      name: 'Crawlix.io',
      url: 'https://crawlix.io',
    },
    offers: [
      {
        '@type': 'Offer',
        name: 'Starter',
        description:
          'Basis-Audit für einfache Websites. 6 Analyse-Module, 30+ Seiten Report, Schema-Analyse, White-Label.',
        price: '400',
        priceCurrency: 'EUR',
        availability: 'https://schema.org/InStock',
        url: 'https://crawlix.io/de/preise',
      },
      {
        '@type': 'Offer',
        name: 'Professional',
        description:
          'Komplett-Audit mit GEO-Check. 8 Analyse-Module, 50+ Seiten Report, Wettbewerbs-Gap-Analyse, 4-Phasen Aktionsplan.',
        price: '600',
        priceCurrency: 'EUR',
        availability: 'https://schema.org/InStock',
        url: 'https://crawlix.io/de/preise',
      },
      {
        '@type': 'Offer',
        name: 'Complete',
        description:
          'Alles inklusive Umsetzungs-Support. Fertiger Code, Implementierungs-Support, 30 Tage Nachbetreuung, Priority 24h Lieferung.',
        price: '1200',
        priceCurrency: 'EUR',
        availability: 'https://schema.org/InStock',
        url: 'https://crawlix.io/de/preise',
      },
    ],
  };

  return (
    <script
      type="application/ld+json"
      dangerouslySetInnerHTML={{ __html: JSON.stringify(schema) }}
    />
  );
}
```

Einbindung in der Preise-Seite:

```tsx
// app/[locale]/preise/page.tsx
import { PricingSchema } from '@/components/seo/PricingSchema';

export default function PreisePage() {
  return (
    <>
      <PricingSchema />
      {/* ... bestehender Content */}
    </>
  );
}
```

Aufwand: 1 Std **Impact:** Rich Results für 3 Pakete in SERPs möglich

---

2.2 BlogPosting Schema vervollständigen

Problem: Bestehendes BlogPosting-Schema hat kein `image` und kein `publisher.logo` – ohne diese zeigt Google **keine** Article Rich Results an.

Fix: Bestehendes BlogPosting-Schema ersetzen. Vermutlich in einer Blog-Layout-Komponente oder per generateMetadata:

```tsx
// components/seo/BlogPostSchema.tsx
interface BlogPostSchemaProps {
  headline: string;
  description: string;
  slug: string;
  datePublished: string;
  dateModified?: string;
  keywords?: string;
  imageUrl?: string; // [TODO: verifizieren] Hero-Bild pro Artikel oder Fallback auf OG-Image
}

export function BlogPostSchema({
  headline,
  description,
  slug,
  datePublished,
  dateModified,
  keywords,
  imageUrl = 'https://crawlix.io/og-image.png', // Fallback
}: BlogPostSchemaProps) {
  const schema = {
    '@context': 'https://schema.org',
    '@type': 'BlogPosting',
    headline,
    description,
    datePublished,
    dateModified: dateModified || datePublished,
    image: {
      '@type': 'ImageObject',
      url: imageUrl,
      width: 1200,
      height: 630,
    },
    author: {
      '@type': 'Person',
      name: 'Lukas Lavicka',
      url: 'https://crawlix.io',
      // [TODO: verifizieren] LinkedIn-URL eintragen:
      sameAs: ['https://www.linkedin.com/in/lukaslavicka'],
    },
    publisher: {
      '@type': 'Organization',
      name: 'Crawlix.io',
      url: 'https://crawlix.io',
      logo: {
        '@type': 'ImageObject',
        url: 'https://crawlix.io/logo.png', // [TODO: verifizieren] Dediziertes Logo (min. 112x112px, quadratisch)
        width: 512,
        height: 512,
      },
    },
    mainEntityOfPage: {
      '@type': 'WebPage',
      '@id': `https://crawlix.io/de/blog/${slug}`,
    },
    url: `https://crawlix.io/de/blog/${slug}`,
    keywords,
    inLanguage: 'de',
    isPartOf: {
      '@type': 'Blog',
      name: 'Crawlix.io Blog',
      url: 'https://crawlix.io/de/blog',
    },
  };
```

```
  return (
    <script
      type="application/ld+json"
      dangerouslySetInnerHTML={{ __html: JSON.stringify(schema) }}
    />
  );
}
```

> **Wichtig:** Es muss ein echtes Logo unter `https://crawlix.io/logo.png` existieren (min. 112x112px, idealerweise 512x512px, quadratisch). Falls kein Logo vorhanden ist, eins erstellen und in `/public/logo.png` ablegen.

Aufwand: 1 Std **Impact:** Article Rich Results in Google aktiviert für alle 10 Blog-Posts

---

2.3 BreadcrumbList Schema global

Problem: Nur `/de/seo-check` hat BreadcrumbList. Fehlt auf Preise, Partner, Blog und allen Blog-Artikeln.

Lösung: Globale BreadcrumbList-Komponente:

```tsx
// components/seo/BreadcrumbSchema.tsx
interface BreadcrumbItem {
  name: string;
  url: string;
}

export function BreadcrumbSchema({ items }: { items: BreadcrumbItem[] }) {
  const schema = {
    '@context': 'https://schema.org',
    '@type': 'BreadcrumbList',
    itemListElement: items.map((item, index) => ({
      '@type': 'ListItem',
      position: index + 1,
      name: item.name,
      item: item.url,
    })),
  };

  return (
    <script
      type="application/ld+json"
      dangerouslySetInnerHTML={{ __html: JSON.stringify(schema) }}
    />
  );
}
```

Einbindung pro Seitentyp:

```
// app/[locale]/preise/page.tsx
<BreadcrumbSchema
  items={[
    { name: 'Home', url: 'https://crawlix.io/de' },
    { name: 'Preise', url: 'https://crawlix.io/de/preise' },
  ]}
/>

// app/[locale]/partner/page.tsx
<BreadcrumbSchema
  items={[
    { name: 'Home', url: 'https://crawlix.io/de' },
    { name: 'Partner', url: 'https://crawlix.io/de/partner' },
  ]}
/>

// app/[locale]/blog/[slug]/page.tsx
<BreadcrumbSchema
  items={[
    { name: 'Home', url: 'https://crawlix.io/de' },
    { name: 'Blog', url: 'https://crawlix.io/de/blog' },
    { name: headline, url: `https://crawlix.io/de/blog/${slug}` },
  ]}
/>
```

Aufwand: 1 Std **Impact:** Breadcrumb Rich Results auf allen Unterseiten

---

2.4 Organization Schema bereinigen

Problem: Das Organization-Schema wird auf JEDER Seite identisch ausgegeben (~500 Bytes pro Seite). Zusätzlich: `alternateName: "meinBetrieb online"` ist veraltet, `logo` zeigt auf og-image.png statt ein echtes Logo, `email` ist eine Gmail-Adresse.

Fix: Organization Schema NUR auf der Homepage ausgeben und bereinigen:

```
// components/seo/OrganizationSchema.tsx -- NUR auf Homepage einbinden
export function OrganizationSchema() {
  const schema = {
    '@context': 'https://schema.org',
    '@type': 'Organization',
    name: 'Crawlix.io',
    url: 'https://crawlix.io',
    logo: {
      '@type': 'ImageObject',
      url: 'https://crawlix.io/logo.png', // [TODO: verifizieren] Dediziertes Logo
      width: 512,
      height: 512,
    },
    description:
      'White-Label SEO-Audit-Tool für Agenturen und Entwickler. 8-Modul-Reports mit AI-Readiness-Check in 48
        Stunden.',
    founder: {
      '@type': 'Person',
      name: 'Lukas Lavicka',
    },
    foundingDate: '2026', // [TODO: verifizieren]
    address: {
      '@type': 'PostalAddress',
      streetAddress: 'Ludwig-Klapp-Str. 5',
      addressLocality: 'Berlin',
      postalCode: '12437',
      addressCountry: 'DE',
    },
    contactPoint: {
      '@type': 'ContactPoint',
      telephone: '+49-1515-0333110',
      email: 'info@crawlix.io', // [TODO: verifizieren] Domain-Mail statt Gmail
      contactType: 'customer service',
      availableLanguage: ['German', 'English', 'Czech'],
    },
    sameAs: [
      // [TODO: verifizieren] Echte Profile-URLs eintragen:
      'https://www.linkedin.com/company/crawlix',
      'https://github.com/crawlix',
    ],
  };

  return (
    <script
      type="application/ld+json"
      dangerouslySetInnerHTML={{ __html: JSON.stringify(schema) }}
    />
  );
}
```

Was entfernt/geändert wurde: - `alternateName: "meinBetrieb online"` –> entfernt (Legacy-Name) - `logo` –> dediziertes Logo statt og-image.png - `email` –> `info@crawlix.io` statt Gmail - `sameAs` –> Social-Media-Profile ergänzt - `description` + `foundingDate` –> neu hinzugefügt

Aus dem globalen Layout entfernen (bisher wird Organization auf jeder Seite ausgegeben):

```
// app/[locale]/layout.tsx
// Organization-Schema-Block HIER ENTFERNEN
// Stattdessen nur auf der Homepage einbinden (app/[locale]/page.tsx)
```

Aufwand: 30 Min **Impact:** Brand-Konsistenz, E-E-A-T Signale, ~500 Bytes weniger pro Unterseite

---

2.5 WebApplication Schema auf Homepage

Problem: Das Hauptprodukt ist nirgends als Schema beschrieben. Die Homepage hat Organization + WebSite + WebPage, aber kein Product/App-Schema.

```tsx
// components/seo/HomeAppSchema.tsx
export function HomeAppSchema() {
  const schema = {
    '@context': 'https://schema.org',
    '@type': 'WebApplication',
    name: 'Crawlix SEO-Audit',
    description:
      '8-Modul SEO-Audit mit GEO/AI-Readiness Check. White-Label Reports für Agenturen in 48 Stunden.',
    url: 'https://crawlix.io',
    applicationCategory: 'BusinessApplication',
    operatingSystem: 'Web',
    browserRequirements: 'Moderner Webbrowser',
    offers: {
      '@type': 'AggregateOffer',
      lowPrice: '400',
      highPrice: '1200',
      priceCurrency: 'EUR',
      offerCount: '3',
      url: 'https://crawlix.io/de/preise',
    },
    author: {
      '@type': 'Organization',
      name: 'Crawlix.io',
      url: 'https://crawlix.io',
    },
    featureList: [
      '8-Modul SEO-Audit',
      'AI-Readiness Check (ChatGPT, Perplexity, Google AI Overviews)',
      'White-Label Branding',
      'Fertiger Code für Fixes',
      'Priorisierter Aktionsplan',
    ],
    inLanguage: ['de', 'en', 'cs'],
  };

  return (
    <script
      type="application/ld+json"
      dangerouslySetInnerHTML={{ __html: JSON.stringify(schema) }}
    />
  );
}
```

Aufwand: 30 Min **Impact:** Software Rich Results in SERPs möglich

---

2.6 SoftwareApplication auf /seo-check korrigieren

Problem: Umlaut-Fehler (“Pruefe” statt “Prüfe”), falscher @type (SoftwareApplication statt WebApplication), Name enthält Pipe-Branding.

Bestehendes Schema ersetzen:

```
{
  "@context": "https://schema.org",
  "@type": "WebApplication",
  "name": "Crawlix SEO-Schnellcheck",
  "description": "Kostenloser SEO-Check: Prüfe deine Website in 60 Sekunden auf 6 SEO-Faktoren.",
  "url": "https://crawlix.io/de/seo-check",
  "applicationCategory": "BusinessApplication",
  "operatingSystem": "Web",
  "offers": {
    "@type": "Offer",
    "price": "0",
    "priceCurrency": "EUR",
    "availability": "https://schema.org/InStock"
  },
  "author": {
    "@type": "Organization",
    "name": "Crawlix.io",
    "url": "https://crawlix.io"
  },
  "inLanguage": "de"
}
```

Änderungen: 1. `SoftwareApplication` –> `WebApplication` 2. “Pruefe” –> “Prüfe” (Umlaut-Fix) 3. Name: “SEO-Schnellcheck | Crawlix.io” –> “Crawlix SEO-Schnellcheck” (kein Pipe-Branding) 4. `availability` ergänzt

Aufwand: 15 Min

2.7 H1-Fix – Nur 1 H1 pro Seite

Problem lt. Action Plan: Homepage hatte 19+ H1-Tags, Partner 9+, Preise 3.

> **Hinweis:** Der Deep-Audit (Remaining-Audits) zeigt Homepage mit 1 H1 und alle Unterseiten ebenfalls mit je 1 H1. Wenn die H1-Mehrfach-Probleme bereits gefixt wurden, diesen Punkt überspringen. Andernfalls:

Prüfung:

```
# Im Browser DevTools Console:
document.querySelectorAll('h1').length
// Erwartung: 1 auf jeder Seite

# Alle H1-Elemente anzeigen:
document.querySelectorAll('h1').forEach(h => console.log(h.textContent))
```

Fix-Prinzip: Nur die Hauptüberschrift als `<h1>`, alle anderen auf `<h2>` oder `<h3>` ändern. Häufige Ursache in Next.js: Komponenten die ihr eigenes `<h1>` mitbringen (z.B. Feature-Cards, Pricing-Tabellen, Hero-Sections).

Aufwand: 1-2 Std (je nach aktuellem Stand) **Impact:** On-Page SEO signifikant verbessert

---

2.8 og:url auf allen Seiten ergänzen

Problem: `og:url` fehlt auf allen geprüften Seiten. Social Sharing könnte falsche URLs anzeigen.

Fix in generateMetadata:

```
// app/[locale]/layout.tsx oder pro Seite
export async function generateMetadata({ params }: { params: { locale: string } }) {
  const { locale } = params;

  return {
    // ... bestehende metadata
    openGraph: {
      // ... bestehende OG-Tags
      url: `https://crawlix.io/${locale}`,
    },
  };
}
```

Für Unterseiten jeweils die korrekte URL setzen:

```
// app/[locale]/preise/page.tsx
openGraph: {
  url: `https://crawlix.io/${locale}/preise`,
}

// app/[locale]/blog/[slug]/page.tsx
openGraph: {
  url: `https://crawlix.io/${locale}/blog/${slug}`,
}
```

Aufwand: 15 Min **Impact:** Korrekte Canonical-URL bei Social Shares

---

2.9 x-default hreflang auf allen Subpages

Problem: 42 von 45 URLs in der Sitemap fehlt x-default. Die HTTP-Header haben es korrekt, die Sitemap nicht – inkonsistent.

Fix: In der Sitemap-Generierung (`app/sitemap.ts`) das x-default ergänzen (siehe Code-Block unter 1.1 oben – dort ist x-default bereits enthalten).

Zusätzlich im HTML-Head, falls hreflang auch dort gesetzt wird:

```ts
// lib/hreflang.ts (Utility)
export function generateHreflang(path: string) {
  const base = 'https://crawlix.io';
  return {
    de: `${base}/de${path}`,
    en: `${base}/en${path}`,
    cs: `${base}/cs${path}`,
    'x-default': `${base}${path}`,
  };
}

// Verwendung in generateMetadata:
export async function generateMetadata({ params }) {
  return {
    alternates: {
      languages: generateHreflang('/preise'),
    },
  };
}
```

Aufwand: 1 Std **Impact:** Konsistente Hreflang-Signale für Suchmaschinen

---

Phase 3: Performance & Technical (Woche 3-4)

> **Erwarteter Score-Sprung: ~75 –> ~82/100**

3.1 LCP optimieren (4.3s –> <2.5s Mobile)

Problem: LCP 4.325ms auf Mobile – fast doppelt über Googles 2.5s-Schwellenwert.

Identifizierte Bottlenecks:

BOTTLENECK	EINSPARUNG	FIX
Redirect-Chain (Root –> /de)	845ms	Nicht vermeidbar auf Vercel
Unused JavaScript	600ms	Bundle-Analyse + Tree-Shaking
Google Fonts render-blocking	~200ms	Fonts selbst hosten
LCP-Element nicht priorisiert	~300ms	fetchPriority + Preload

3.1.1 LCP-Element priorisieren

```
// Für das Hero-Element (Text oder Bild) im Above-the-fold-Bereich:
// Falls es ein Bild ist:
import Image from 'next/image';

<Image
  src="/hero-image.webp"
  alt="Crawlix SEO-Audit Dashboard"
  width={1200}
  height={630}
  priority  // Deaktiviert Lazy Loading, setzt fetchPriority="high"
  sizes="100vw"
/>

// Falls der LCP-Trigger ein Text-Element ist:
// Sicherstellen dass die Fonts schnell laden (siehe 3.1.2)
```

3.1.2 Google Fonts selbst hosten

Vorteile: Spart DNS-Lookup + CORS-Overhead + DSGVO-konform

```
# Fonts herunterladen (google-webfonts-helper oder fontsource):
npm install @fontsource/inter @fontsource/plus-jakarta-sans
```

```
// app/layout.tsx
import '@fontsource/inter/400.css';
import '@fontsource/inter/500.css';
import '@fontsource/inter/600.css';
import '@fontsource/inter/700.css';
import '@fontsource/plus-jakarta-sans/500.css';
import '@fontsource/plus-jakarta-sans/600.css';
import '@fontsource/plus-jakarta-sans/700.css';
import '@fontsource/plus-jakarta-sans/800.css';
```

Dann entfernen aus dem HTML-Head:

```
<!-- Diese Zeilen entfernen: -->
<link rel="preload" href="https://fonts.googleapis.com/css2?family=Inter..." as="style"/>
<link rel="preconnect" href="https://fonts.googleapis.com"/>
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin="anonymous"/>
<link href="https://fonts.googleapis.com/css2?family=..." rel="stylesheet"/>
```

Alternativ mit Next.js built-in Font Optimization (empfohlen):

```tsx
// app/layout.tsx
import { Inter, Plus_Jakarta_Sans } from 'next/font/google';

const inter = Inter({
  subsets: ['latin'],
  display: 'swap',
  variable: '--font-inter',
});

const plusJakartaSans = Plus_Jakarta_Sans({
  subsets: ['latin'],
  display: 'swap',
  variable: '--font-plus-jakarta',
});

export default function RootLayout({ children }) {
  return (
    <html className={`${inter.variable} ${plusJakartaSans.variable}`}>
      <body>{children}</body>
    </html>
  );
}
```

> `next/font` hosted die Fonts automatisch selbst (kein externer Request mehr) und optimiert mit `size-adjust` für CLS.

3.1.3 JavaScript-Bundle analysieren

```
# Bundle-Analyse starten:
ANALYZE=true next build

# Oder mit dem offiziellen Plugin:
npm install @next/bundle-analyzer

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

module.exports = withBundleAnalyzer({
  // ... bestehende Config
});
```

Prüfen ob Turbopack für Production optimal ist (aktuell erkannt: `turbopack-*.js` Chunks).

Aufwand: 2-3 Std (gesamt für 3.1) **Impact:** Mobile Performance 79 –> 85+, LCP von 4.3s auf unter 2.5s

---

3.2 Security Headers vervollständigen

Problem: `Permissions-Policy` und `X-Powered-By` Header.

Fix in `next.config.js`:

```
// next.config.js (oder next.config.mjs)
/** @type {import('next').NextConfig} */
const nextConfig = {
  // X-Powered-By deaktivieren
  poweredByHeader: false,

  // Security Headers
  async headers() {
    return [
      {
        source: '/(.*)',
        headers: [
          {
            key: 'Permissions-Policy',
            value: 'camera=(), microphone=(), geolocation=(), browsing-topics=()',
          },
          // Bestehende Header beibehalten:
          // Strict-Transport-Security, Content-Security-Policy,
          // X-Frame-Options, X-Content-Type-Options, Referrer-Policy
          // sind bereits korrekt konfiguriert
        ],
      },
    ];
  },
};

module.exports = nextConfig;
```

Verifikation:

```
curl -I https://crawlix.io/de 2>/dev/null | grep -i -E "permissions-policy|x-powered-by"
# Erwartet:
# permissions-policy: camera=(), microphone=(), geolocation=(), browsing-topics=()
# KEIN x-powered-by Header
```

Aufwand: 15 Min **Impact:** Security Score: 85 –> 95/100

3.3 Internal Linking fixen

Problem 1: Kaputte Links im Blog (fehlender Locale-Prefix)

6 Links im Blog-Artikel “Was kostet ein SEO Audit?” verwenden die falsche URL-Struktur:

KAPUTT	KORREKT
`/blog/white-label-seo-audit-tool-vergleich`	`/de/blog/white-label-seo-audit-tool-vergleich`
`/blog/strukturierte-daten-testen-guide`	`/de/blog/strukturierte-daten-testen-guide`
`/blog/chatgpt-seo-sichtbarkeit`	`/de/blog/chatgpt-seo-sichtbarkeit`
`/preise`	`/de/preise`
`/partner`	`/de/partner`
`/blog/seo-relaunch-checkliste`	`/de/blog/seo-relaunch-checkliste`

Fix: In den Blog-MDX/Content-Dateien die Links korrigieren. Besser: Einen Link-Utility nutzen der automatisch den Locale-Prefix setzt:

```
// lib/link.ts
export function localizedHref(path: string, locale: string = 'de'): string {
  // Sicherstellen dass der Locale-Prefix vorhanden ist
  if (path.startsWith(`/${locale}/`) || path.startsWith(`/${locale}`)) {
    return path;
  }
  return `/${locale}${path.startsWith('/') ? '' : '/'}${path}`;
}
```

Problem 2: Verwaiste Seiten – seo-check und preise nicht in Navigation

SEITE	STATUS	FIX
`/de/seo-check`	Nirgends verlinkt	In Navigation aufnehmen
`/de/preise`	Nur via /partner#formular erreichbar	“Preis” Nav-Link auf `/de/preise` statt `/#pricing` ändern

> **Fix in der Navigation-Komponente:**

```
// components/Navigation.tsx (oder wo die Nav definiert ist)
const navItems = [
  // VORHER: { href: '/#pricing', label: 'Preis' }
  // NACHHER:
  { href: '/de/preise', label: 'Preise' },

  // NEU hinzufügen:
  { href: '/de/seo-check', label: 'SEO-Check' },

  // ... bestehende Items
];
```

Problem 3: Hub-Spoke Verlinkung fehlt

Cross-Links zwischen Service-Seiten und Blog einbauen:

VON	NACH	ANCHOR-TEXT
`/de/preise`	`/de/blog/was-kostet-seo-audit`	“Detaillierter Preisvergleich im Blog”
`/de/preise`	`/de/partner`	“Partner-Konditionen ansehen”
`/de/partner`	`/de/preise`	“Alle Pakete und Preise”
`/de/seo-check`	`/de/preise`	“Vollständigen Audit bestellen”
`/de/blog/*`	`/de/seo-check`	“Kostenloser Schnellcheck”

Aufwand: 2 Std **Impact:** Internal Linking Score: 3/10 –> 7/10, verwaiste Seiten eliminiert

---

3.4 Color-Contrast Fix (WCAG 1.4.3)

Problem: 4 Elemente mit unzureichendem Kontrast. Minimum-Ratio: 4.5:1 für normalen Text.

Diagnose – betroffene Elemente identifizieren:

```
# Lighthouse im Browser:
# DevTools > Lighthouse > Accessibility
# Oder axe-core im Terminal:
npx @axe-core/cli https://crawlix.io/de
```

Fix-Prinzip: Farbwerte in der Tailwind-Config oder den CSS-Variablen anpassen, bis das Kontrastverhältnis >= 4.5:1 ist.

Kontrast prüfen: https://webaim.org/resources/contrastchecker/

Aufwand: 30 Min **Impact:** Accessibility Score: 91 –> 95+

3.5 Umlaut-Encoding auf SEO-Check-Seite fixen

Problem: Content zeigt “Pruefe”, “Ueberschriften”, “haeufigsten” statt korrekte Umlaute. Das deutet auf ein Encoding-Problem im CMS/Template.

Wahrscheinliche Ursache: Die Übersetzungsdatei oder Content-Datei für die SEO-Check-Seite wurde mit ASCII-Encoding geschrieben statt UTF-8.

Fix: In der Übersetzungsdatei (z.B. `messages/de.json`) oder im Content die ASCII-Äquivalente ersetzen:

```
"Pruefe"        -->  "Prüfe"
"Ueberschriften" -->  "Überschriften"
"haeufigsten"   -->  "häufigsten"
```

Sicherstellen dass die Datei als UTF-8 gespeichert ist:

```
# Encoding prüfen:
file -I messages/de.json
# Erwartet: charset=utf-8
```

Aufwand: 15 Min

Phase 4: Images & Content (Monat 2)

> **Erwarteter Score-Sprung: ~82 –> ~88/100**

4.1 Bilder hinzufügen (aktuell: ZERO)

Problem: Die gesamte Website hat **kein einziges `<img>`-Tag**. 14 Inline-SVGs, aber keine Rasterbilder. Das bedeutet: - Zero Google Images Präsenz - Keine Screenshots des Produkts (Conversion-Problem für ein SaaS-Tool) - BlogPosting-Schema ohne `image` = keine Article Rich Results

Empfohlene Bilder:

SEITE	BILD	ALT-TEXT	PRIORITÄT
Homepage Hero	Screenshot des Audit-Dashboards	“Crawlix SEO-Audit Dashboard mit Score-Übersicht”	HOCH
/de/preise	Beispiel-Report Cover	“Crawlix White-Label SEO-Report Beispiel”	HOCH
Blog-Artikel	Pro Artikel ein Hero-Image	Beschreibend pro Thema	HOCH
/de/partner	White-Label Report Mockup	“White-Label SEO-Audit Report mit eigenem Branding”	MITTEL

Next.js Image Component verwenden:

```
// Beispiel: Hero-Image auf Homepage
import Image from 'next/image';

export function HeroImage() {
  return (
    <Image
      src="/images/audit-dashboard.webp"
      alt="Crawlix SEO-Audit Dashboard mit Score-Übersicht und 8-Modul-Analyse"
      width={1200}
      height={675}
      priority  // Above-the-fold = priority
      sizes="(max-width: 768px) 100vw, (max-width: 1200px) 80vw, 1200px"
      className="rounded-lg shadow-lg"
    />
  );
}
```

Bild-Optimierung in next.config.js:

```
// next.config.js
const nextConfig = {
  images: {
    formats: ['image/avif', 'image/webp'],
    deviceSizes: [640, 750, 828, 1080, 1200, 1920],
    imageSizes: [16, 32, 48, 64, 96, 128, 256, 384],
  },
};
```

> Next.js `<Image>` liefert automatisch: WebP/AVIF-Konvertierung, responsive srcset, Lazy Loading (außer bei `priority`), CLS-Prevention durch feste Dimensionen.

SVG Accessibility – alle dekorativen SVGs:

```
// Dekorative SVGs:
<svg aria-hidden="true" focusable="false" /* ... */ />

// Informative SVGs (wenn sie Inhalt vermitteln):
<svg role="img" aria-label="Beschreibung des Icons">
  <title>Beschreibung</title>
  {/* ... */}
</svg>
```

Aufwand: 4-8 Std (Screenshots erstellen + einbauen) **Impact:** Image SEO aktiviert, Google Images Präsenz, Article Rich Results

---

4.2 Author-Box Komponente für Blog

Problem: Kein Autoren-Profil, keine E-E-A-T-Signale auf Blog-Posts.

```
// components/blog/AuthorBox.tsx
import Image from 'next/image';

interface AuthorBoxProps {
  name: string;
  role: string;
  bio: string;
  avatarUrl: string;
  linkedinUrl?: string;
  websiteUrl?: string;
}

export function AuthorBox({
  name = 'Lukas Lavicka',
  role = 'Developer & SEO-Analyst',
  bio = 'Gründer von crawlix.io. Baut SEO-Audit-Tools für Agenturen, die ihre Kunden mit datengetriebenen Reports
überzeugen wollen.',
  avatarUrl = '/images/authors/lukas-lavicka.webp',
  linkedinUrl = 'https://www.linkedin.com/in/lukaslavicka', // [TODO: verifizieren]
  websiteUrl = 'https://crawlix.io',
}: Partial<AuthorBoxProps>) {
  return (
    <aside className="mt-12 border-t border-gray-200 pt-8">
      <div className="flex items-start gap-4">
        <Image
          src={avatarUrl}
          alt={`Portrait von ${name}`}
          width={80}
          height={80}
          className="rounded-full"
        />
        <div>
          <p className="font-semibold text-lg">{name}</p>
          <p className="text-sm text-gray-500">{role}</p>
          <p className="mt-2 text-gray-700">{bio}</p>
          <div className="mt-3 flex gap-3">
            {linkedinUrl && (
              <a
                href={linkedinUrl}
                target="_blank"
                rel="noopener noreferrer"
                className="text-sm text-blue-600 hover:underline"
              >
                LinkedIn
              </a>
            )}
            {websiteUrl && (
              <a
                href={websiteUrl}
                className="text-sm text-blue-600 hover:underline"
              >
                Website
              </a>
            )}
          </div>
        </div>
      </div>
    </aside>
  );
}
```

Einbindung in Blog-Layout:

```
// app/[locale]/blog/[slug]/page.tsx
import { AuthorBox } from '@/components/blog/AuthorBox';

export default function BlogPost({ params }) {
  return (
    <article>
      {/* Blog-Content */}
      <AuthorBox />
    </article>
  );
}
```

Aufwand: 2 Std **Impact:** E-E-A-T Signale gestärkt, Autoren-Attribution für AI-Engines

---

4.3 Blog-Veröffentlichungsdaten staffeln

Problem: Alle 10 Blog-Posts haben identisches `datePublished: 2026-03-25` – Bulk-Publishing-Signal.

Fix: Publikationsdaten über mehrere Wochen verteilen:

```
was-kostet-seo-audit          --> 2026-02-10
white-label-seo-audit-tool    --> 2026-02-17
chatgpt-seo-sichtbarkeit      --> 2026-02-24
onpage-seo-checkliste          --> 2026-03-03
strukturierte-daten-testen     --> 2026-03-10
seo-relaunch-checkliste        --> 2026-03-14
seo-agentur-tools-vergleich    --> 2026-03-17
local-seo-audit-guide          --> 2026-03-20
technisches-seo-basics         --> 2026-03-23
geo-ai-search-guide            --> 2026-03-25
```

In den Blog-Content-Dateien oder der Datenquelle (CMS, MDX-Frontmatter) das `date`-Feld anpassen. Gleichzeitig `lastmod` / `dateModified` auf das tatsächliche letzte Änderungsdatum setzen.

Aufwand: 30 Min **Impact:** Natürlicheres Publishing-Pattern, bessere Freshness-Signale

---

4.4 Thin Content auf /de/preise erweitern

Problem: 370 Wörter – deutlich unter dem empfohlenen Minimum (600-800 Wörter) für eine kommerzielle Seite.

Empfohlene Ergänzungen: - Abschnitt “Was ist in jedem Paket enthalten?” mit Detail-Auflistung - Cross-Link zum Blog-Artikel “Was kostet ein SEO Audit?” - Vergleichstabelle “crawlix vs. manuelles Audit” - FAQ-Sektion (falls nicht vorhanden)

Aufwand: 2 Std (Content schreiben + einbauen) **Impact:** Content Score der Preisseite deutlich verbessert

---

4.5 OG-Tags Fix auf SEO-Check-Seite

Problem: Die SEO-Check-Seite nutzt Homepage OG-Tags als Fallback: - OG Title: “SEO-Audit der auch prüft ob AI dich findet | Crawlix.io” (= Homepage!)

Fix:

```
// app/[locale]/seo-check/page.tsx
export const metadata = {
  title: 'Kostenloser SEO-Schnellcheck | Crawlix.io',
  description:
    'Prüfe deine Website in 60 Sekunden auf 6 SEO-Faktoren. Score 0-100 mit konkreten Handlungsempfehlungen.',
  openGraph: {
    title: 'Kostenloser SEO-Schnellcheck | Crawlix.io',
    description:
      'Prüfe deine Website in 60 Sekunden auf 6 SEO-Faktoren. Score 0-100 mit konkreten Handlungsempfehlungen.',
    url: 'https://crawlix.io/de/seo-check',
    // [TODO: verifizieren] Eigenes OG-Image für SEO-Check erstellen
    images: [{ url: 'https://crawlix.io/og-seo-check.png', width: 1200, height: 630 }],
  },
};
```

Aufwand: 15 Min

Implementation Checklist

#	TASK	PHASE	ABHÄNGIGKEIT	AUFWAND	STATUS
1	EN/CS Blog-URLs noindex setzen	P1	Keine	30 Min	[ ]
2	llms.txt erstellen + deployen	P1	Keine	15 Min	[ ]
3	MwSt-Widerspruch Text fixen	P1	Keine	10 Min	[ ]
4	i18n-Keys in Locale-Datei ergänzen	P1	Keine	30 Min	[ ]
5	Starter Seitenzahl-Widerspruch fixen	P1	Keine	5 Min	[ ]
6	“98% Genauigkeit” entfernen/belegen	P1	Keine	5 Min	[ ]
7	Partner-Seite: unique Title+Description	P1	Keine	15 Min	[ ]
8	Product/Offer Schema Preise-Seite	P2	Keine	1 Std	[ ]
9	BlogPosting Schema vervollständigen	P2	Logo-Datei (#15)	1 Std	[ ]
10	BreadcrumbList global	P2	Keine	1 Std	[ ]
11	Organization Schema bereinigen	P2	Logo-Datei (#15)	30 Min	[ ]
12	WebApplication Schema Homepage	P2	Keine	30 Min	[ ]
13	SoftwareApplication Fix /seo-check	P2	Keine	15 Min	[ ]
14	H1-Tags prüfen und fixen	P2	Keine	1-2 Std	[ ]
15	Logo-Datei erstellen (512x512px)	P2	Keine	30 Min	[ ]

#	TASK	PHASE	ABHÄNGIGKEIT	AUFWAND	STATUS
16	og:url auf allen Seiten	P2	Keine	15 Min	[ ]
17	x-default hreflang auf Subpages	P2	Keine	1 Std	[ ]
18	LCP optimieren (Fonts, Bundle)	P3	Keine	2-3 Std	[ ]
19	Permissions-Policy Header	P3	Keine	15 Min	[ ]
20	poweredByHeader: false	P3	Keine	2 Min	[ ]
21	Kaputte Blog-Links fixen (6 Links)	P3	Keine	30 Min	[ ]
22	Navigation: /seo-check + /preise	P3	Keine	15 Min	[ ]
23	Hub-Spoke Cross-Links einbauen	P3	#22	1 Std	[ ]
24	Color-Contrast Fix (4 Elemente)	P3	Keine	30 Min	[ ]
25	Umlaut-Encoding /seo-check	P3	Keine	15 Min	[ ]
26	Bilder/Screenshots hinzufügen	P4	Bilder erstellen	4-8 Std	[ ]
27	Author-Box Komponente	P4	Autor-Foto	2 Std	[ ]
28	Blog-Daten staffeln	P4	Keine	30 Min	[ ]
29	Thin Content /preise erweitern	P4	Keine	2 Std	[ ]
30	OG-Tags Fix /seo-check	P4	Keine	15 Min	[ ]
31	SVG Accessibility (aria-hidden)	P4	Keine	30 Min	[ ]

Erwartete Score-Entwicklung

PHASE	MASSNAHMEN	SCORE	NOTE
Aktuell	–	57/100	F
Nach Phase 1	#1-7	~65/100	D
Nach Phase 2	#8-17	~75/100	C
Nach Phase 3	#18-25	~82/100	B-
Nach Phase 4	#26-31	~88/100	B+

Validierung nach Implementation

Schema validieren

```
# Google Rich Results Test:
# https://search.google.com/test/rich-results?url=https://crawlix.io/de/preise

# Schema.org Validator:
# https://validator.schema.org/

# Oder per CLI (wenn vorhanden):
npx schema-dts-gen --url https://crawlix.io/de/preise
```

Lighthouse lokal testen

```
# Mobile Performance:
npx lighthouse https://crawlix.io/de --view --preset=perf --form-factor=mobile

# Full Audit:
npx lighthouse https://crawlix.io/de --view
```

hreflang validieren

```
# Alle hreflang-Tags einer Seite anzeigen:
curl -s https://crawlix.io/de/preise | grep -i hreflang

# HTTP-Header hreflang prüfen:
curl -I https://crawlix.io/de/preise 2>/dev/null | grep -i "link.*hreflang"
```

Sitemap prüfen

```
# Sitemap herunterladen und URLs zählen:
curl -s https://crawlix.io/sitemap.xml | grep -c "<url>"
# Erwartet nach Phase 1: 25 (statt 45, wenn EN/CS Blog entfernt)
```

Nicht empfohlene Schema-Typen

TYP	GRUND
~~**HowTo**~~	Deprecated seit September 2023 – Rich Results entfernt
~~**FAQPage**~~	Nur für Government/Healthcare empfohlen (eingeschränkt seit August 2023)
~~**SpecialAnnouncement**~~	Deprecated seit Juli 2025
~~**Review/AggregateRating**~~ **(self-serving)**	Nur für echte Nutzerbewertungen, nicht eigene Testimonials

Anhang: Dateistruktur-Übersicht

```
public/
├── llms.txt                    # NEU (Phase 1)
├── logo.png                    # NEU (Phase 2) -- 512x512px, quadratisch
├── images/
│   ├── audit-dashboard.webp    # NEU (Phase 4) -- Hero-Image
│   ├── report-example.webp     # NEU (Phase 4) -- Beispiel-Report
│   └── authors/
│       └── lukas-lavicka.webp  # NEU (Phase 4) -- Autoren-Foto

components/
├── seo/
│   ├── PricingSchema.tsx       # NEU (Phase 2)
│   ├── BlogPostSchema.tsx      # NEU (Phase 2)
│   ├── BreadcrumbSchema.tsx    # NEU (Phase 2)
│   ├── OrganizationSchema.tsx  # NEU (Phase 2)
│   └── HomeAppSchema.tsx       # NEU (Phase 2)
├── blog/
│   └── AuthorBox.tsx           # NEU (Phase 4)
└── Navigation.tsx              # BEARBEITEN (Phase 3)

app/
├── [locale]/
│   ├── layout.tsx              # BEARBEITEN (Font-Optimierung, Organization entfernen)
│   ├── page.tsx                # BEARBEITEN (H1, Schema, Hero-Image)
│   ├── preise/page.tsx         # BEARBEITEN (MwSt-Text, Schema, Content)
│   ├── partner/page.tsx        # BEARBEITEN (Title, Description)
│   ├── seo-check/page.tsx      # BEARBEITEN (Schema, OG-Tags, Umlaute)
│   └── blog/[slug]/page.tsx    # BEARBEITEN (BlogPosting Schema, AuthorBox, Links)
├── sitemap.ts                  # BEARBEITEN (EN/CS Blog entfernen)
├── middleware.ts               # NEU/BEARBEITEN (noindex für EN/CS Blog)

next.config.js                  # BEARBEITEN (poweredByHeader, Permissions-Policy, Images)
messages/de.json                # BEARBEITEN (i18n-Keys für Partner-Seite)
```

Report erstellt am 2026-03-31 auf Basis von 8 Audit-Quellen. Alle Code-Snippets sind copy-paste-ready für Next.js 15 (App Router) auf Vercel. [TODO: verifizieren]-Marker kennzeichnen Werte die vor Deployment geprüft werden müssen.