Multi-Project Support Implementation Guide

Overzicht

Dit document beschrijft de implementatie van multi-project support in Writgo.nl, waarmee één account meerdere websites/projecten kan beheren.

Wat is geïmplementeerd?

1. Database Schema ✅

- **Migratie:** `/supabase/migrations/20251212_multi_project_support.sql`
- Toegevoegd aan Project tabel:
- `status` (active/inactive/archived)
- `settings` (JSONB voor project-specifieke instellingen)
- `createdAt` en `updatedAt` timestamps
- Alle content tabellen hebben nu `projectId` foreign keys:
- `BlogPost`
- `ContentPlan`
- `TopicalAuthorityMap`
- `SocialMediaStrategy`
- `WebsiteAnalysis`
- `AutopilotConfig`
- RLS policies per project voor data isolatie

2. Context Management ✅

- **Bestand:** `/lib/contexts/ProjectContext.tsx`
- Functionaliteit:
- `currentProject`: Huidige actieve project
- `projects`: Lijst van alle projecten
- `switchProject(id)`: Wissel tussen projecten
- `addProject(data)`: Voeg nieuw project toe
- `updateProject(id, data)`: Update project
- `deleteProject(id)`: Verwijder project
- LocalStorage voor laatste project onthouden

3. API Endpoints ✅

- **GET** `/api/admin/projects` - Haal alle projecten op
- **POST** `/api/admin/projects` - Maak nieuw project
- **GET** `/api/admin/projects/[id]` - Haal specifiek project op
- **PUT** `/api/admin/projects/[id]` - Update project
- **DELETE** `/api/admin/projects/[id]` - Verwijder project

4. UI Components ✅

- **ProjectSwitcher** (`/components/project/ProjectSwitcher.tsx`)

- Dropdown in sidebar
- Toon huidige project
- Wissel tussen projecten
- Voeg nieuw project toe
- Ga naar project management
- **AddProjectDialog** (`/components/project/AddProjectDialog.tsx`)
- Modal voor nieuw project toevoegen
- Validatie van naam, URL, beschrijving
- **Project Management Page** (`/app/admin/projects/page.tsx`)
- Grid view van alle projecten
- Activeer, bewerk, verwijder projecten
- Visual indicator voor actief project

5. Layout Integratie ✅

- ProjectProvider toegevoegd aan `/app/admin/layout.tsx`
- ProjectSwitcher toegevoegd aan sidebar
- “Projecten Beheer” toegevoegd aan navigatie

Hoe te gebruiken in je code?

1. Gebruik de ProjectContext in Components

```
'use client';

import { useProject } from '@/lib/contexts/ProjectContext';

export function MyComponent() {
  const { currentProject, projects, switchProject } = useProject();

  if (!currentProject) {
    return <div>Geen project geselecteerd</div>;
  }

  return (
    <div>
      <h1>Huidige Project: {currentProject.name}</h1>
      <p>Website: {currentProject.websiteUrl}</p>

      {/* Dropdown om te switchen */}
      <select onChange={(e) => switchProject(e.target.value)}>
        {projects.map(p => (
          <option key={p.id} value={p.id}>
            {p.name}
          </option>
        ))}
      </select>
    </div>
  );
}
```

2. Data Fetching met ProjectId

Voorbeeld: Blog posts ophalen voor huidige project

```
'use client';

import { useEffect, useState } from 'react';
import { useProject } from '@/lib/contexts/ProjectContext';

export function BlogPostList() {
  const { currentProject } = useProject();
  const [posts, setPosts] = useState([]);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    if (!currentProject) return;

    fetchPosts();

    // Listen for project changes
    const handleProjectChange = () => fetchPosts();
    window.addEventListener('projectChanged', handleProjectChange);

    return () => {
      window.removeEventListener('projectChanged', handleProjectChange);
    };
  }, [currentProject]);

  const fetchPosts = async () => {
    try {
      setLoading(true);
      const response = await fetch(
        `/api/admin/blog?projectId=${currentProject.id}`
      );
      const data = await response.json();
      setPosts(data.posts || []);
    } catch (error) {
      console.error('Failed to fetch posts:', error);
    } finally {
      setLoading(false);
    }
  };

  if (loading) return <div>Laden...</div>;

  return (
    <div>
      <h2>Blog Posts voor {currentProject.name}</h2>
      {posts.map(post => (
        <div key={post.id}>{post.title}</div>
      ))}
    </div>
  );
}
```

3. API Endpoint Updates

Voorbeeld: Update blog API om projectId te gebruiken

```
// app/api/admin/blog/route.ts

export async function GET(request: Request) {
  try {
    const session = await getServerSession(authOptions);

    if (!session || !session.user) {
      return NextResponse.json({ error: 'Unauthorized' }, { status: 401 });
    }

    // Haal projectId uit query parameters
    const { searchParams } = new URL(request.url);
    const projectId = searchParams.get('projectId');

    if (!projectId) {
      return NextResponse.json(
        { error: 'Project ID required' },
        { status: 400 }
      );
    }

    // Verificeer dat project bij deze user hoort
    const project = await prisma.project.findUnique({
      where: { id: projectId }
    });

    if (!project || project.clientId !== session.user.id) {
      return NextResponse.json(
        { error: 'Unauthorized' },
        { status: 403 }
      );
    }

    // Haal alleen posts voor dit project
    const posts = await prisma.blogPost.findMany({
      where: { projectId },
      orderBy: { createdAt: 'desc' }
    });

    return NextResponse.json({ posts });
  } catch (error) {
    console.error('Failed to fetch posts:', error);
    return NextResponse.json(
      { error: 'Failed to fetch posts' },
      { status: 500 }
    );
  }
}

export async function POST(request: Request) {
  try {
    const session = await getServerSession(authOptions);

    if (!session || !session.user) {
      return NextResponse.json({ error: 'Unauthorized' }, { status: 401 });
    }

    const body = await request.json();
    const { title, content, projectId } = body;

    // Validatie
    if (!projectId) {
```

```
      return NextResponse.json(
        { error: 'Project ID required' },
        { status: 400 }
      );
    }

    // Verificeer project ownership
    const project = await prisma.project.findUnique({
      where: { id: projectId }
    });

    if (!project || project.clientId !== session.user.id) {
      return NextResponse.json(
        { error: 'Unauthorized' },
        { status: 403 }
      );
    }

    // Maak post met projectId
    const post = await prisma.blogPost.create({
      data: {
        projectId,
        title,
        content,
        // ... andere velden
      }
    });

    return NextResponse.json({ post }, { status: 201 });
  } catch (error) {
    console.error('Failed to create post:', error);
    return NextResponse.json(
      { error: 'Failed to create post' },
      { status: 500 }
    );
  }
}
```

Belangrijke Opmerkingen

1. Data Isolatie

- Elke project heeft zijn eigen content
- BlogPosts, ContentPlans, TopicalMaps, etc. zijn allemaal project-specifiek
- RLS policies zorgen ervoor dat clients alleen hun eigen project data kunnen zien

2. Project Switching Event

- Wanneer gebruiker van project wisselt, wordt een custom event gefired:
  `javascript`
  `window.dispatchEvent(new CustomEvent('projectChanged', {`
  `detail: { projectId }`
  `}));`
- Luister naar dit event om je component data te refreshen

3. Default Project

- Bij het aanmaken van een nieuwe client wordt automatisch een default project aangemaakt
- Dit project is gemarkeerd met `isPrimary: true`

- Zie: `/app/api/admin/clients/route.ts`

4. Empty States

- Handel situaties af waar geen project geselecteerd is
- Toon friendly messages en call-to-actions om project toe te voegen

Testing Checklist

- [ ] Nieuw project aanmaken werkt
- [ ] Projecten switcher toont alle projecten
- [ ] Switchen tussen projecten werkt
- [ ] Content is project-specifiek (geen data leaks)
- [ ] Project bewerken werkt
- [ ] Project verwijderen werkt (met bevestiging)
- [ ] Laatste project wordt onthouden na page refresh
- [ ] Empty state toont correct bericht
- [ ] API endpoints valideren projectId correct
- [ ] RLS policies blokkeren unauthorized access

Volgende Stappen

Migreer Bestaande Endpoints

De volgende endpoints moeten nog gemigreerd worden om projectId te gebruiken:

1. **Blog Management**
   - `/api/admin/blog/content-plan/*`
   - `/api/admin/blog/topical-map/*`
2. **Social Media**
   - `/api/admin/social/*`
3. **Website Analysis**
   - `/api/admin/analyzer/*`

Data Migration Script

Voor bestaande data zonder projectId:

```
-- Voor elke client zonder projectId in content
UPDATE "BlogPost"
SET "projectId" = (
  SELECT id FROM "Project"
  WHERE "clientId" = "BlogPost"."clientId"
  AND "isPrimary" = true
  LIMIT 1
)
WHERE "projectId" IS NULL;

-- Herhaal voor andere content tabellen
```

Support

Voor vragen of problemen, zie:
- Database schema: `/supabase/migrations/20251212_multi_project_support.sql`
- Context API: `/lib/contexts/ProjectContext.tsx`
- Voorbeeld implementatie: `/app/admin/projects/page.tsx`