Guide för flerspråkig implementation i Next.js med App Router

Denna guide beskriver hur du kan implementera flerspråkighet i din AAIMS-applikation med Next.js och App Router.

Innehållsförteckning

Steg 1: Konfigurera App Router för flerspråkighet

Börja med att skapa en struktur som stödjer dynamiska språkparametrar:

```
app/
├── [lang]/
│   ├── layout.tsx
│   ├── page.tsx
│   ├── meetings/
│   │   └── page.tsx
│   ├── transcripts/
│   │   └── page.tsx
│   └── settings/
│       └── page.tsx
└── layout.tsx (root layout)
```

Root Layout (app/layout.tsx)

Detta är din grundläggande layout som omdirigerar till standardspråket:

```
 export default function RootLayout() {
  // Omdirigera till standardspråket (svenska)
  return <RedirectToDefaultLanguage defaultLang="sv" />;
}

function RedirectToDefaultLanguage({ defaultLang }: { defaultLang: string }) {
  // Implementera omdirigering till /sv eller använd språkdetektering
  // Detta kan göras med middleware eller klient-side redirect
}
```

Språkspecifik Layout (app/[lang]/layout.tsx)

```
 import { i18n } from '@/config/i18n';

export async function generateStaticParams() {
  return i18n.locales.map(locale => ({ lang: locale }));
}

export default function LocaleLayout({
  children,
  params: { lang }
}: {
  children: React.ReactNode;
  params: { lang: string };
}) {
  return (
    <html lang={lang}>
      <body>
        {/* Språkväljare och gemensamma komponenter */}
        <LanguageSwitcher currentLang={lang} />
        {children}
      </body>
    </html>
  );
}
```

Steg 2: Skapa översättningsfiler

Skapa en mapp för dina översättningar:

```
/src/translations/
  /sv/
```

```
    common.json
    meetings.json
  /en/
    common.json
    meetings.json
```

Exempel på innehåll i `common.json`:

```
// sv/common.json
{
  "nav": {
    "home": "Hem",
    "meetings": "Möten",
    "transcripts": "Transkriberingar",
    "settings": "Inställningar"
  },
  "actions": {
    "create": "Skapa",
    "edit": "Redigera",
    "delete": "Ta bort",
    "save": "Spara"
  }
}

// en/common.json
{
  "nav": {
    "home": "Home",
    "meetings": "Meetings",
    "transcripts": "Transcripts",
    "settings": "Settings"
  },
  "actions": {
    "create": "Create",
    "edit": "Edit",
    "delete": "Delete",
    "save": "Save"
  }
}
```

Steg 3: Skapa en översättningsfunktion

```
// src/utils/translations.ts
import { cache } from 'react';

export const getTranslations = cache(async (lang: string, namespace: string)
=> {
```

```
  try {
    return (await import(`@/translations/${lang}/${namespace}.json`)).default;
  } catch (error) {
    console.error(`Failed to load translations for ${lang}/${namespace}`, error);
    return {};
  }
});

export async function getTranslation(
  lang: string,
  namespace: string,
  key: string
) {
  const translations = await getTranslations(lang, namespace);
  return key.split('.').reduce((obj, k) => obj?.[k], translations) || key;
}
```

Steg 4: Skapa en Translator-komponent

```
// src/components/Translator.tsx
import { getTranslation } from '@/utils/translations';

export async function T({
  lang,
  ns = 'common',
  k,
}: {
  lang: string;
  ns?: string;
  k: string;
}) {
  const translation = await getTranslation(lang, ns, k);
  return <>{translation}</>;
}
```

Steg 5: Implementera språkväljare

```
// src/components/LanguageSwitcher.tsx
'use client';

import { usePathname, useRouter } from 'next/navigation';
import { i18n } from '@/config/i18n';

export default function LanguageSwitcher({ currentLang }: { currentLang:
string }) {
  const pathname = usePathname();
```

```
  const router = useRouter();

  const handleLanguageChange = (newLang: string) => {
    // Byt ut nuvarande språk i URL:en
    const newPathname = pathname.replace(`/${currentLang}`, `/${newLang}
`);
    router.push(newPathname);
  };

  return (
    <div className="flex space-x-2">
      {i18n.locales.map((locale) => (
        <button
          key={locale}
          onClick={() => handleLanguageChange(locale)}
          className={` px-3 py-1 rounded ${
            currentLang === locale
              ? 'bg-blue-500 text-white'
              : 'bg-gray-200 hover:bg-gray-300'
          }`}
        >
          {locale.toUpperCase()}
        </button>
      ))}
    </div>
  );
}
```

Steg 6: Konfigurera i18n-inställningar

```
// src/config/i18n.ts
export const i18n = {
  defaultLocale: 'sv',
  locales: ['sv', 'en'],
};

export const getLocaleFromPath = (path: string) => {
  const locale = path.split('/')[1];
  return i18n.locales.includes(locale) ? locale : i18n.defaultLocale;
};
```

Steg 7: Använda översättningar i dina sidor

```
// app/[lang]/page.tsx
import { T } from '@/components/Translator';
```

```
export default function HomePage({ params: { lang } }: { params: { lang: string } }) {
  return (
    <div className="container mx-auto p-4">
      <h1 className="text-3xl font-bold mb-4">
        <T lang={lang} k="home.title" />
      </h1>
      <p className="mb-4">
        <T lang={lang} k="home.description" />
      </p>

      {/* Resten av din hemsida */}
    </div>
  );
}
```

Steg 8: Hantera språkspecifika AI-transkriberingar

För att hantera språkspecifika AI-transkriberingar, kan du skapa en funktion som väljer rätt modell baserat på språk:

```
// src/utils/transcription.ts
export async function transcribeMeeting(audioUrl: string, language: string) {
  // Välj rätt modell baserat på språk
  const speechModel = language === 'sv' ? 'nano' : 'default';

  // Anropa AssemblyAI med rätt parametrar
  const response = await fetch('https://api.assemblyai.com/v2/transcript', {
    method: 'POST',
    headers: {
      'Authorization': process.env.ASSEMBLY_AI_KEY,
      'Content-Type': 'application/json',
    },
    body: JSON.stringify({
      audio_url: audioUrl,
      language_code: language,
      speech_model: speechModel, // Använd nano för svenska
    }),
  });

  return response.json();
}
```

Steg 9: Implementera metadata för SEO

```tsx
// app/[lang]/layout.tsx
import { Metadata } from 'next';
import { getTranslations } from '@/utils/translations';

export async function generateMetadata({
  params: { lang }
}: {
  params: { lang: string }
}): Promise<Metadata> {
  const t = await getTranslations(lang, 'common');

  return {
    title: t.metadata?.title || 'Meeting AI',
    description: t.metadata?.description || 'AI Meeting Secretary',
    alternates: {
      languages: {
        'sv': '/sv',
        'en': '/en',
      },
    },
  };
}
```

Steg 10: Hantera flerspråkiga routes

För att hantera dynamiska routes på olika språk, kan du använda parametrar:

```tsx
// app/[lang]/meetings/[id]/page.tsx
import { T } from '@/components/Translator';

export default function MeetingDetailPage({
  params: { lang, id }
}: {
  params: { lang: string; id: string }
}) {
  return (
    <div>
      <h1>
        <T lang={lang} k="meetings.detail.title" />: {id}
      </h1>
      {/* Resten av din mötesdetalj-sida */}
    </div>
  );
}
```

Steg 11: Middleware för språkdetektering (valfritt)

Om du vill automatiskt omdirigera användare baserat på deras webbläsarspråk:

```tsx
// middleware.ts
import { NextResponse } from 'next/server';
import type { NextRequest } from 'next/server';
import { i18n } from '@/config/i18n';

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

  // Kontrollera om pathname redan har ett språkprefix
  const pathnameHasLocale = i18n.locales.some(
    locale => pathname.startsWith(`/${locale}/`) || pathname === `/${locale}`
  );

  if (pathnameHasLocale) return;

  // Detektera användarens språk från Accept-Language header
  const acceptLanguage = request.headers.get('accept-language') || '';
  const userLocale = acceptLanguage
    .split(',')
    .map(lang => lang.split(';')[0].trim())
    .find(lang => i18n.locales.includes(lang.substring(0, 2)));

  // Använd detekterat språk eller standardspråk
  const locale = userLocale ? userLocale.substring(0, 2) : i18n.defaultLocale;

  // Omdirigera till korrekt språkversion
  return NextResponse.redirect(
    new URL(`/${locale}${pathname === '/' ? '' : pathname}`, request.url)
  );
}

export const config = {
  matcher: ['/((?!api|_next/static|_next/image|favicon.ico).*)'],
};
```

Utöka med fler språk

När du vill lägga till fler språk (t.ex. tyska och franska), behöver du:

1. Uppdatera `i18n.locales` i `src/config/i18n.ts`: `tsx export const i18n = { defaultLocale: 'sv', locales: ['sv', 'en', 'de', 'fr'], };`

2. Skapa nya översättningsfiler: `/src/translations/ /de/ common.json meetings.json /fr/ common.json meetings.json`

3. Uppdatera metadata i `app/[lang]/layout.tsx`: `tsx alternates: { languages: { 'sv': '/sv', 'en': '/en', 'de': '/de', 'fr': '/fr', }, },`

Bästa praxis för flerspråkighet

1. **Håll översättningarna organiserade**: Dela upp översättningar i logiska namespaces (common, meetings, etc.)
2. **Använd fallback-värden**: Se till att det alltid finns ett fallback-värde om en översättning saknas
3. **Testa alla språk**: Testa regelbundet alla språkversioner av din app
4. **Använd språkspecifika formatering**: För datum, nummer och valutor, använd språkspecifika formateringar
5. **Optimera för SEO**: Använd hreflang-taggar och språkspecifika metadata
6. **Överväg innehållsöversättning**: För dynamiskt innehåll, överväg att använda översättningstjänster eller AI

Språkspecifika överväganden för AAIMS

Svenska

- För AssemblyAI-transkribering, använd alltid `speech_model: 'nano'` med `language_code: 'sv'`
- Formatera datum enligt svenskt format (YYYY-MM-DD)
- Använd svenska tidsformat (24-timmarsklocka)

Engelska

- För AssemblyAI-transkribering, använd standardmodellen
- Formatera datum enligt engelskt format (MM/DD/YYYY eller DD/MM/YYYY beroende på målgrupp)
- Överväg både amerikansk och brittisk engelska om relevant

Tyska och Franska (framtida expansion)

- Undersök språkspecifika AI-modeller för transkribering
- Implementera korrekt formatering för datum, tid och nummer
- Överväg kulturella skillnader i användargränssnittet

Slutsats

Genom att följa denna guide kan du skapa en robust flerspråkig struktur för din AAIMS-applikation med Next.js och App Router. Strukturen är skalbar och kan enkelt utökas med fler språk i framtiden.